МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)
20 травня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 — Комп'ютерних наук,
освітньо-наукової програми «Інформатика»
на тему «Інформаційна технологія проєктування віртуального навчального тренажера на основі скінченного автомата з додатковою пам'яттю»
здобувача групи ІН.м-21н Сивоконя Вадима Володимировича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Вадим СИВОКОНЬ
(підпис)

Керівник,
доцент кафедри,
кандидат фізико-математичних наук Оксана ШОВКОПЛЯС _____
(підпис)

Суми — 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр

зі спеціальності 122 — Комп'ютерних наук, освітньо-наукової програми «Інформатика»
здобувача групи ІН.м-21н Сивоконя Вадима Володимировича

1. Тема роботи: *«Інформаційна технологія проєктування віртуального навчального тренажера на основі скінченного автомата з додатковою пам'яттю»*
затверджую наказом по СумДУ від «08» березня 2024 р. № *0234-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 21 травня 2024 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз предметної області, постановка задачі. 2) Вибір моделей та методів дослідження. 3) Розроблення комплекту матеріалів для проведення освітнього експерименту. 4) Аналіз результатів дослідження.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «1» квітня 2024 р.

Завдання прийняв до виконання _____ (підпис) Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз предметної області, формулювання постановки задачі дослідження*	01.04.2024–04.04.2024	Виконано
2	*Вибір методів дослідження, моделювання та проєктування інформаційної технології*	05.04.2024–06.04.2024	Виконано
3	*Розроблення матеріалів для експерименту*	07.04.2024–20.04.2024	Виконано
4	*Проведення освітнього експерименту*	21.04.2024–09.05.2024	Виконано
5	*Аналіз результатів дослідження, оформлення розрахунково-пояснювальної записки*	10.05.2024–18.05.2024	Виконано

Здобувач вищої освіти _____ (підпис) Керівник _____ (підпис)

АНОТАЦІЯ

Записка: 267 стор., 62 рис., 23 табл., 11 додатків, 105 джерел.

Обґрунтування актуальності теми роботи — розробляється велика кількість навчальних інтерактивних матеріалів на основі новітніх технологій. Використання сучасних інтерфейсів взаємодії користувача з комп'ютером підвищує ефективність освіти, утім стримуючим фактором залишається недостатня гнучкість процесів формування завдань в існуючих віртуальних тренажерах. Логіка переходів спирається на поточний стан та керуючий вплив. Неврахування попередньої взаємодії негативно впливає на спроможність інтерактивного навчального матеріалу до повноцінного висвітлення теми, тому постає задача розроблення технології, яка б дозволила врахувати передісторію роботи.

Об'єкт дослідження — інтерактивне навчання в закладах вищої освіти.

Мета роботи — розробити інформаційну технологію проєктування віртуального навчального тренажера на основі скінченного автомата з додатковою пам'яттю.

Методи дослідження — методи теорії автоматів, метод класифікацій, метод декомпозиції, гіпотетичний метод, метод проб, розроблення вебдодатків, методи математичної статистики, соціологічні методи.

Результати — розроблений із використанням нової технології віртуальний тренажер на тему «Транспортні задачі» впроваджено до освітнього процесу в рамках дисципліни «Математичні методи дослідження операцій», яка викладається на кафедрі комп'ютерних наук Сумського державного університету.

ВІРТУАЛЬНИЙ ТРЕНАЖЕР, ІНТЕРАКТИВНЕ НАВЧАННЯ, ПРАКТИЧНЕ ЗАВДАННЯ, СКІНЧЕННИЙ АВТОМАТ ІЗ ДОДАТКОВОЮ ПАМ'ЯТТЮ, ТРАНСПОРТНА ЗАДАЧА, ФРЕЙМВОРК

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API — інтерфейс програмування застосунку

DOM — об’єктна модель документа

DOT — мова опису графів

ESM — модуль ECMAScript

IDE — інтегроване середовище розроблення

JSX — JavaScript XML

SMIL — мова синхронізованої інтеграції засобів мультимедіа

TSX — TypeScript XML

XML — розширювана мова розмітки

ВСТУП

Актуальність. Ефективність електронного навчання стримується низькою гнучкістю процесів формування завдань в існуючих віртуальних тренажерах, а наукові розробки закордонних (К. Чонг (K. Chong), Б. Марсена (B. Marcen), Е. Коррала (E. Corral), У. Росміаті (U. Rosmiati), Б. Еріксон (B. Ericson), В. Нур Хідаята (W. Nur Hidayat), Б. Ларраги (B. Larraga) та ін.) та вітчизняних (В. О. Остапенка, Н. А. Доценко, Р. Г. Валєєва, В. М. Михалевича, І. В. Перегняка та ін.) учених не дозволяють враховувати в моделях передісторію взаємодії користувача з навчальним об'єктом. У зв'язку з цим постає задача розроблення інформаційної технології проєктування, що базується на моделі скінченного автомата з додатковою пам'яттю.

Об'єкт дослідження. Інтерактивне навчання в закладах вищої освіти.

Предмет дослідження. Інформаційна технологія проєктування віртуального навчального тренажера на основі скінченного автомата з додатковою пам'яттю.

Мета та завдання дослідження. Метою кваліфікаційної роботи є підвищення гнучкості процесу формування завдань віртуальними навчальними тренажерами шляхом розроблення інформаційної технології проєктування на основі моделі скінченного автомата з додатковою пам'яттю.

Для досягнення зазначеної мети необхідно вирішити такі завдання:

— проаналізувати останні наукові публікації, присвячені використанню інтерактивних навчальних матеріалів в освітньому процесі, та відповідні програмні продукти;
— розробити математичну модель віртуального тренажера на основі скінченного автомата з додатковою пам'яттю;
— розробити графічну нотацію для проєктування логіки переходів між завданнями тренажера на основі нової моделі;

— розробити математичну модель оцінювання навчальних досягнень для тренажера на основі удосконаленої автоматної моделі;
— розробити графічну нотацію для візуалізації процесу нарахування балів за новою математичною моделлю оцінювання;
— розробити програмну реалізацію інформаційної технології проєктування віртуального тренажера на основі скінченного автомата з додатковою пам’яттю;
— розробити основний тренажер на тему «Транспортні задачі» із використанням нової технології;
— розробити альтернативний тренажер на тему «Транспортні задачі» із використанням наявної технології, в основі якої — звичайна автоматна модель;
— розробити інформаційне забезпечення експерименту: пам’ятку, інструкції до тренажерів (основного та альтернативного), поточний та підсумковий тест, а також анкету учасника освітнього експерименту;
— порівняти ефективність основного та альтернативного тренажера шляхом проведення освітнього експерименту.

Зв’язок роботи з науковими програмами, планами, темами. Наукові дослідження за темою кваліфікаційної роботи пов’язані з виконанням кафедрою комп’ютерних наук Сумського державного університету науково-дослідної роботи «Застосування технологій games learning, blended learning, віртуальної та доповненої реальності в навчальному процесі», 2020-2025 рр. (№ ДР 0120U103407).

Суперечність, що вирішується в роботі. Відсутність у наукових публікаціях закордонних (К. Чонг (K. Chong), Б. Марсена (B. Marcen), Е. Коррала (E. Corral), У. Росміаті (U. Rosmiati), Б. Еріксон (B. Ericson), В. Хідаята (W. Hidayat), Б. Ларраги (B. Larraga) та ін.) та вітчизняних (В. О. Остапенка, Н. А. Доценко, Р. Г. Валєєва, В. М. Михалевича, І. В. Перегняка та ін.) дослідників інтерактивного навчання моделі віртуального тренажера, де б формування наступного завдання

міг би залежати не тільки від попереднього й наданої відповіді, а й від передісторії взаємодії користувача з навчальним об’єктом.

Гіпотеза дослідження. Якщо доповнити існуючу автоматну модель віртуального тренажера інструментом додаткової пам’яті, можна забезпечити достатню для потреб практики гнучкість процесу формування завдань, де враховуватиметься не лише поточний стан навчального об’єкта та керувальний вплив користувача, а й попередньо збережені значення.

Наукова новизна. Удосконалено автоматну модель віртуального тренажера шляхом введення додаткової пам’яті, що на відміну від звичайної дозволяє враховувати попередньо збережені значення під час переходів між завданнями інтерактивного навчального об’єкта.

Практичне значення отриманих результатів. Запропонована інформаційна технологія дозволяє проєктувати інтерактивні тренажери з достатньою для потреб практики гнучкістю процесу формування завдань завдяки вдосконаленій автоматній моделі, що додатково передбачає використання раніше збережених до пам’яті значень під час переходів.

Апробація результатів. Основні положення та результати кваліфікаційної роботи висвітлені та обговорені на міжнародних науково-практичних та науково-технічних конференціях: Міжнародній науково-практичній сателітній конференції «Інформаційні технології та впровадження» (Київ, Київський національний університет імені Тараса Шевченка; 2023); Міжнародній науково-технічній конференції студентів та молодих вчених «Інформатика, Математика, Автоматика» (Суми, Сумський державний університет; 2023); Міжнародній науково-технічній конференції студентів та молодих вчених «Інформатика, Математика, Автоматика» (Суми, Сумський державний університет; 2024).

Структура та обсяг роботи. Кваліфікаційна робота складається з анотації, змісту, переліку умовних скорочень, вступу, трьох розділів, висновків, списку використаних джерел і 11 додатків. Праця викладена на 267 сторінках, серед яких анотація на 1 сторінці, зміст на 2 сторінках, перелік умовних скорочень на

1 сторінці, основний текст на 155 сторінках, список використаних джерел із 105 найменувань на 11 сторінках та додатки на 95 сторінках. Робота містить 23 таблиці та 62 рисунки.

Публікації. За темою кваліфікаційної роботи опубліковано 4 праці [1–4], серед яких 1 стаття в науковому фаховому виданні України, 3 тези доповідей у матеріалах та програмах міжнародних науково-практичних та науково-технічних конференцій.

Використання результатів та впровадження. Практичний результат дослідження у вигляді віртуального тренажера на тему «Транспортні задачі», розробленого з використанням нової технології, впроваджено до освітнього процесу з дисципліни «Математичні методи дослідження операцій», що викладається на кафедрі комп’ютерних наук Сумського державного університету (акт впровадження від 13.05.2024 р.).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Сучасний стан та тенденції розвитку інтерактивних навчальних матеріалів

Сучасні навчальні матеріали обмежені за своєю ефективністю. Переважно використовується статична інформація — тексти та зображення. Широкого розповсюджені навчальні відеоматеріали, які сприяють засвоєнню знань завдяки якісній візуалізації та динамічності. Спільний недолік цих типів контенту — недостатня залученість учнів та студентів до освітнього процесу, бо кількість самостійно розв’язаних завдань важливіша за кількість переглянутих прикладів.

Рішення проблеми полягає в застосуванні інтерактивних навчальних матеріалів — сучасного формату подання інформації для організації активної взаємодії. Він передбачає не лише перегляд мультимедіа, а й виконання супровідних динамічних завдань для закріплення знань.

Закордонні та вітчизняні вчені займаються розробленням нових моделей та методів інтерактивного навчання для задоволення зростаючих потреб практики. Про це свідчить гістограма дослідженості проблеми (див. рис. 1.1).

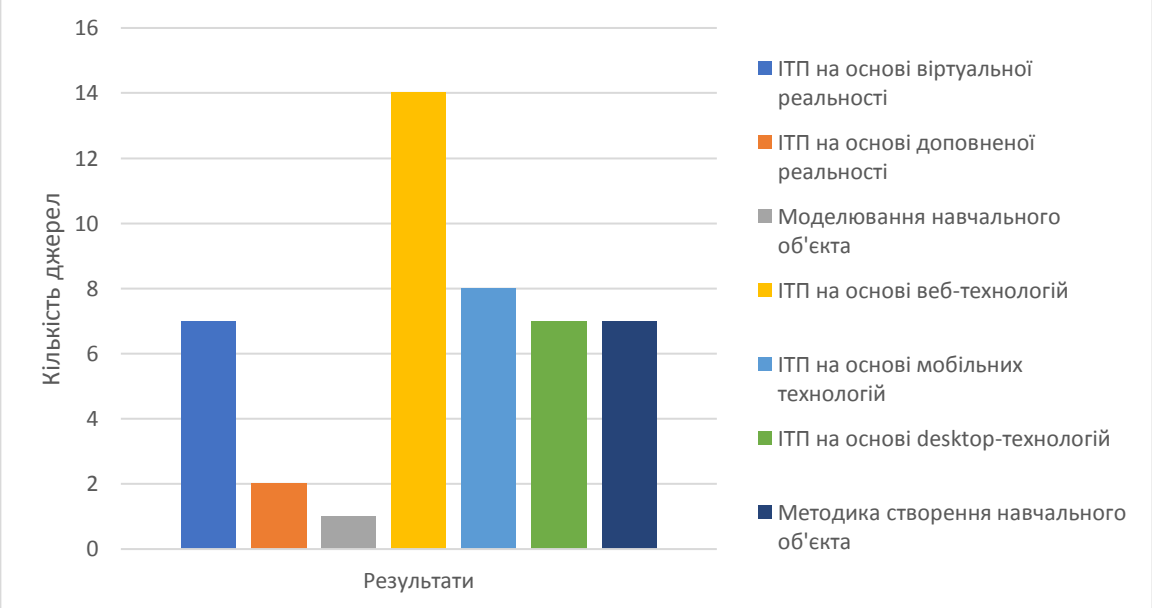


Рисунок 1.1 — Гістограма дослідженості проблеми створення ефективних інтерактивних навчальних матеріалів

Переважна кількість публікацій присвячена експериментальним дослідженням ефективності інформаційних технологій проєктування (ІТП) інтерактивних навчальних об’єктів різного типу, таких як симулятори, тренажери та інтерактивні підручники. Їхня наукова новизна ґрунтується на використанні вебтехнологій [5–18], віртуальної [19–25] та доповнено реальності [26, 27], інструментів мобільної [28–35] та desktop-розробки [36–42], завдяки яким для навчання створюються моделі людино-машинної взаємодії. Меншою мірою представлені теоретичні дослідження з розроблення відповідних методик [43–49]. Тема моделювання, зокрема математичного, залишається однією з найменш досліджених [50].

Використання засобів інтерактивного навчання не обмежується лише галуззю освіти: школами [5, 28–30, 46] та вищими навчальними закладами [6, 11–20, 26, 31, 32, 36–39, 47]. Також є праці, що висвітлюють їх застосування в промисловості [10, 23, 50] та медицині [9, 22] для оволодіння професійними навичками. Крім того є публікації, спрямовані на підвищення ефективності формування інших компетентностей, серед яких навички гри в настільний теніс [24] та практики медитації [25].

1.2 Технології розроблення навчальних симуляторів

Навчальний симулятор — програмний продукт навчального призначення, що складається з трьох основних частин: симуляції математичної моделі, інтерфейсу взаємодії користувача з моделлю та елементів опису [15]. Розроблення цих додатків — один із найбільш розвинених напрямів досліджень. Причиною тому є постійний розвиток технологій віртуальної реальності, у яких учені вбачають інструменти реалізації ефективного інтерфейсу з точки зору ергономічності та продуктивності.

У праці [19] розглядалася задача розроблення симулятора навчального тренажера (стенда) з вивчення гідравліки. Програмна реалізація спиралась на технологію віртуальної реальності Unity3D. Учені застосували об’єктно-

орієнтовану модель симуляції для кожного гідравлічного компонента, щоб відтворити кожен аспект реального лабораторного дослідження. Вони також подбали про звуковий супровід. За результатами проведеного опитування студентів бакалаврату та магістратури більшість залишилися задоволені досвідом використання впровадженого віртуального інструменту.

Своєю чергою автори статті [27] запропонували підхід на основі технології доповненої реальності для вивчення фізики. Їм вдалося реалізувати симуляцію фізики м'яких тіл у реальному часі завдяки застосуванню можливостей графічного процесора з паралельної обробки та Microsoft Kinect V2. 93% студентів відповіли, що бажають бачити розробку в освітньому процесі.

Технології віртуальної та доповненої реальності забезпечують вищий рівень вражень від навчального контенту. Проте, важливо враховувати, що їх впровадження до освітнього процесу може вимагати значних витрат на придбання необхідного обладнання, особливо у випадках, коли потрібно організувати навчання для цілої групи.

Учені в праці [9] скористалися значно доступнішим вебстеком технологій (HTML, CSS, JavaScript — для frontend-розробки; Node.js, MySQL, Docker — для реалізації backend-частини), щоб створити симулятор із травматології. Принцип його функціонування: тренер налаштовує параметри травми віртуального пацієнта, після чого призначає виконавця завдання. Під час роботи додатку візуалізується пацієнт, його показники життєдіяльності та перелік можливих дій. Під час виконання симулятора реєструється послідовність виконаних дій, витрати часу та вплив на стан пацієнта. За результатами пілотного дослідження встановлено, що такий симулятор створює позитивний вплив на тривалість лікування травм.

Автор праці [18] запропонувала проведення віртуальних практичних робіт та виконання навчально-дослідницьких завдань із хімії із використанням симуляторів проєкту PhET Interactive Simulations. За проведеним контролем знань встановлено, що виконання практичних завдань на симуляторах сприяє підвищенню якості знань до 72% у порівнянні з 52% у референтних групах.

Отже, ключовою особливістю симуляторів є моделювання неперервних процесів, завдяки чому досягається висока реалістичність. Їх розробка є найбільш складною, оскільки потребує застосування потужного математичного апарату. Крім того не всі процеси є детермінованими. Приклад тому — симуляція стану пацієнта, де реакція організму на однакові дії медичного персоналу є індивідуальною для кожної людини.

Технології віртуальної реальності є найбільш ефективним інструментом створення інтерфейсу взаємодії користувача з навчальним середовищем. Використання якісних 3D-моделей у поєднанні з активізацією сенсорних систем людини забезпечують якісну імерсивність. Стримуючим недоліком є значні витрати на закупівлю обладнання, необхідного для повноцінного навчання.

1.3 Технології розроблення навчальних тренажерів

Навчальний тренажер — комп’ютерна навчальна програма для розвитку умінь та навичок із певної діяльності, а також відповідних здібностей. В основу таких додатків закладається певне завдання. Суть використання тренажера в освітньому процесі — ефективне викладення навчального матеріалу як за критерієм часових витрат, так і за якістю набутих учнями та студентами знань [44]. Схема процесу навчання із його використанням наведена на рисунку 1.2.

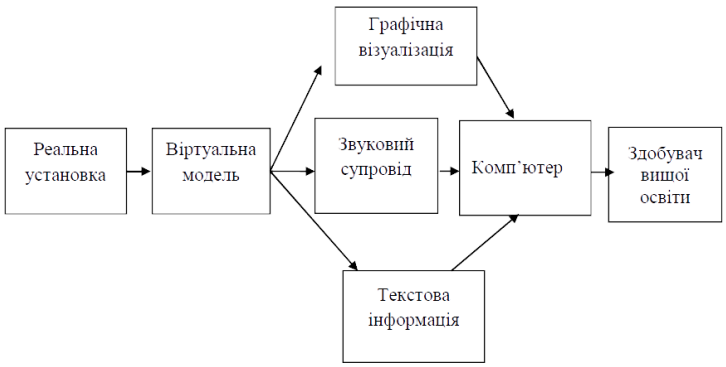


Рисунок 1.2 — Схема процесу навчання із застосуванням навчального комп’ютерного інтерактивного тренажера [45]

У роботі [44] наводять класифікацію інтерактивних навчальних тренажерів. На основі підтримуваних типів завдань виділяють 4 основні класи:

— електронні тестові, в основі яких тестові завдання;
— графічні, що тренують зорове сприйняття та навчають роботі з графічними об’єктами;
— гейміфіковані, що містять ігрове завдання;
— анімовані, які є динамічними — завдання поступово змінюється під час виконання.

Автор у своїй статті [50] виокремив кілька важливих принципів, що використовуються під час створення навчальних тренажерів:

— принцип нелінійності сюжету, коли користувач може впливати на процес навчання;
— принцип різноманітності рішень, що передбачає можливість наявності кількох розв’язків одного завдання;
— принцип незакріпленого порядку розв’язків, що означає можливість обрання порядку виконання завдань навчального об’єкта.

Також у роботі [50] запропонована модифікована модель віртуального тренажера, сформульована в термінах теорії автоматів.

У статті [22] розглядається інструмент для тренування медичних бригад парамедиків, які перші прибувають на місце, де відбулася надзвичайна ситуація з великою кількістю постраждалих. Тріаж — процедура сортування жертв за ступенем необхідності надання невідкладної допомоги. Правильне її застосування дозволяє врятувати життя людей.

Розробники реалізували тренажер на основі технологій віртуальної реальності. Перед навчанням є можливість налаштувати умови, у яких працюватимуть парамедики, а саме: локацію (у приміщенні або на вулиці), кількість жертв (від 1 до 6), стан жертв (рівень серцебиття, наявність кровотеч та ін.), кількість перехожих, наявність відволікаючого шуму від навколишнього середовища, сирен, гелікоптера, а також туману. На початку вбудована перевірка наявності вмінь

виконати ті чи інші дії у віртуальному середовищі, після чого розпочинається тренування. Світ відкритий — можна підходити до будь-якого постраждалого та виконувати доступні дії, але головна умова — керуватися методом тріажу. Проходити тренажер може як одна людина, так і група. Форма звітності — створення аудіозапису після класифікації стану та надання необхідної допомоги.

Дослідження відбувалося в три етапи:

— брифінг, на якому учасники могли ознайомитися з протоколами проведення тріажу та отримати інформацію щодо роботи в тренажері;
— тренування у віртуальній реальності;
— дебрифінг, де надавався зворотний зв’язок щодо успішності проходження завдань у тренажері.

У праці [26] вирішувалася задача розроблення тренажера на основі доповненої реальності для студентів-архітекторів. На основі розпізнавання зображень-маркерів програмний продукт будував 3D-моделі, які можна масштабувати та переміщувати. Експериментальним шляхом доведена ефективність впровадження інструменту з точки зору навчальних досягнень та рівня мотивації. Перевагою такого продукту можна вважати підвищення зацікавленості до навчального контенту: допитливість спонукає до активного дослідження. Недоліком підходу є відсутність завдань, які треба вирішувати користувачу в додатку.

Автори статті [7] запропонували підхід до створення імерсивного вебтренажера шляхом використання панорамних фотографій. Простий метод створення віртуального середовища здатний пришвидшити розробку інтерактивних матеріалів для забезпечення потреб освітнього процесу. З іншого боку недоліком є переходи у вигляді телепортацій, які доводиться супроводжувати спецефектами, щоб приховати різку зміну фотографій.

У статті [28] реалізований механізм надсилання запиту про допомогу під час виконання гейміфікованого навчального тренажера. За результатами експерименту учні, що займалися із використанням мобільного додатку з таким механізмом, показали кращі результати під час оцінювання рівня знань та мотивації

до навчання. Очевидним недоліком є залучення людини, що означає вищий рівень відповідальності, складність надання допомоги кільком учням одночасно. Більш того такий тренажер не працюватиме повноцінно в будь-який зручний час. На тлі інших розробок характерною перевагою є використання простої процедури генерації завдань у рамках різних етапів тренажера з вивчення дробів, що зумовлено використанням часових обмежень на проходження.

Раніше в роботі [47] запропонований підхід до створення навчальних Maple тренажерів (НМТ). Автори керувалися принципом нових задач, ідея якого — уникнення часових витрат на набуття навичок обчислень, які можна виконати за допомогою системи комп'ютерної математики (у даному випадку Maple). Тренажери на основі цього підходу являють собою процедури, що виводять опис покрокового розв'язання математичної задачі для вказаних значень параметрів. У такий спосіб отримується план для окремого випадку, що дуже схоже на вебкалькулятори, які формують інструкцію з розв'язання певних задач, наприклад, транспортних [51]. Підставляючи різні значення параметрів можна слідкувати, як змінюється алгоритм розв'язку. Проте така організація навчання не підходить для оцінювання знань, бо не можна сформувати перелік завдань, які б студент мав розв'язати. Більш того, щоб з'ясувати усі можливі сценарії розв'язання задачі треба досліджувати код процедури, а тому — мати знання програмування, що обмежує застосовність тренажерів такого типу в освітньому процесі.

Отже, для розроблення навчальних тренажерів є велика кількість інструментів, починаючи із засобів desktop-розробки й закінчуючи технологіями віртуальної реальності. Процес розв'язання задач характеризується дискретністю — розглядаються окремі кроки. Відповідно вимоги до математичного апарату значно нижчі в порівнянні з симуляторами.

Спільним недоліком існуючих розробок є проста процедура формування завдань. Майже не використовуються переваги динамічної генерації прикладів, що ймовірно зумовлено додатковою складністю аналізу можливих випадкових

ситуацій, із якими міг би зіштовхнутися користувач. Видача наступного завдання або спирається на попереднє, або зовсім незалежна.

1.4 Технології розроблення інтерактивних підручників

Інтерактивний підручник — електронний підручник із інтерактивними елементами, що використовуються для поліпшення сприйняття інформації.

У роботі [36] запропонований підхід до створення презентацій з математики на основі ефектів zoom in та zoom out, які стали доступні не лише в Prezi (де вважалися «візитною карткою»), а й у більш поширеному офісному додатку PowerPoint. Ці візуальні засоби дозволяють створювати ієрархічні переходи вглибину між слайдами, завдяки чому можна легко створити mind-карту навчальних ресурсів, за якою учням та студентам буде цікаво рухатися. Стримуючім фактором залишається версія PowerPoint 2019 та вище, починаючи з якої розглянуті ефекти стають доступні.

Ще одна desktop-розробка висвітлена в роботі [40], що являє собою віртуальний світ, де надається можливість ознайомлення з культурною спадщиною Пакистану, зокрема залізничною станцією в місті Банну. Імерсивність забезпечується завдяки можливості керування аватаром (персонажем) у цьому додатку. Для зручності реалізовані додаткові засоби переміщень, такі як політ та телепортація. Програмна реалізація спирається на OpenSimulator фреймворк. У порівнянні зі звичайними текстами, фотографіями, відео та 3D-моделями без інтерфейсу для взаємодії інтерактивний програмний продукт вигідно вирізняється. Недоліком є недостатня деталізація моделей пам’яток культури. Утім експеримент показав суттєвий позитивний ефект на рівень знак учасників.

Стаття [8] присвячена вебплатформі для онлайн інтерактивних підручників. Особливістю запропонованого сервісу є можливість викладення теоретичного матеріалу разом із супровідними невеличкими питаннями для закріплення знань. Надаваними перевагами для студентів є інтерактивність, варіативність типів завдань, миттєвий зворотний зв’язок та безоплатність. Увагу авторів

контенту своєю чергою привертає легкість публікації навчальних матеріалів та стабільність платформи.

У роботі [13] вирішувалася задача підвищення залученості студентів до вивчення обчислювального мислення за допомогою використання інтерактивних обчислювальних блокнотів. Під ними автори дослідження мали на увазі продукт, дещо схожий на Jupiter Notebook, але із елементами гейміфікації (кількість очок, спроб) — для підвищення мотивації. Студенти можуть писати та виконувати код. Є можливість попросити підказку, але це призводить до зниження балів, щоб спонукати студентів до самостійності під час розв’язання задач.

Отже, розглянутий напрям є найбільш обмеженим за використовуваними технологіями: немає сенсу у створенні інтерактивних підручників, взаємодіяти із якими можна було б у віртуальній реальності. Мобільна розробка не користується попитом, оскільки динамічність значно простіше реалізувати на основі веб-технологій, аніж створювати окремий додаток. Наявність desktop-реалізацій пов’язана з існуванням інструментів: програми для підготовки презентацій Power Point 2019+ для роботи [36] та фреймворку OpenSimulator для статті [40].

Не всі розглянуті розробки є підручниками в класичному розумінні, але за своєю ідеєю вони надзвичайно схожі: надається простір для активного дослідження, хоч це презентація із mind-картою в Power Point, хоч відкритий віртуальний світ для перегляду пам’яток культури.

Недоліками напрямку є низька складність завдань, оскільки в їх основу не закладаються математичні моделі як такі. Утім інтерактивні підручники легко створюються завдяки існуючим технологіям.

1.5 Постановка задачі дослідження

За результатами аналізу предметної області за актуальними науково-технічними напрямами розроблення моделей та методів створення навчальних симуляторів, тренажерів та підручників можна зробити такі висновки:

— ефективність існуючих навчальних об’єктів стримується використовуваними простими процесами формування завдань, які, якщо й передбачають розгалуження, але без врахування попередньої історії взаємодії;
— процедура динамічної генерації якщо й використовується, проте не здатна забезпечити значну варіативність завдань;
— сучасні технології віртуальної реальності дозволяють забезпечити ефективну реалізацію людино-машинної взаємодії, однак не є доступними для широкого впровадження до освітнього процесу, що зумовлює популярність інших інструментів для створення навчальних об’єктів: засоби веб, мобільної та desktop розроблення.

Виходячи з цього можна стверджувати, що розроблення інформаційної технології проєктування віртуального тренажера на основі скінченного автомата з додатковою пам’яттю є актуальною проблемою. Її вирішення дозволить створювати ефективніші навчальні об’єкти, здатні забезпечити більш повне моделювання практичних задач. Результати дослідження доповнять існуючу систему знань про створення інтерактивних навчальних матеріалів (див. табл. 1.1).

Формулювання завдань, які необхідно вирішити в рамках кваліфікаційної роботи:

— розробити математичну модель віртуального тренажера на основі скінченного автомата з додатковою пам’яттю;
— розробити графічну нотацію для проєктування логіки переходів між завданнями тренажера на основі нової моделі;
— розробити математичну модель оцінювання навчальних досягнень для тренажера на основі удосконаленої автоматної моделі;
— розробити графічну нотацію для візуалізації процесу нарахування балів за новою математичною моделлю оцінювання;
— розробити програмну реалізацію інформаційної технології проєктування віртуального тренажера на основі скінченного автомата з додатковою пам’яттю;

— розробити основний тренажер на тему «Транспортні задачі» із використанням нової технології;
— розробити альтернативний тренажер на тему «Транспортні задачі» із використанням наявної технології, в основі якої — звичайна автоматна модель;
— розробити інформаційне забезпечення експерименту: пам’ятку, інструкції до тренажерів (основного та альтернативного), поточний та підсумковий тест, а також анкету учасника освітнього експерименту;
— порівняти ефективність основного та альтернативного тренажера шляхом проведення освітнього експерименту.

Таблиця 1.1 — Місце дослідження кваліфікаційної роботи в комплексі споріднених проблем

		Об’єкти досліджень		
		Розроблення навчальних симуляторів	Розроблення навчальних тренажерів	Розроблення інтерактивних підручників
Предмети досліджень	Інформаційна технологія проєктування навчального об’єкта на основі віртуальної реальності	Х. Ассаф	С. Шан, У. Гулець, Ф. Кассола, Т. Боне, Х. Шень, Р. Р. Фейнберг, П. Фогт	
	Інформаційна технологія проєктування навчального об’єкта на основі доповненої реальності	Н. Д. Сунг	А. С. Рієра	
	Інформаційна технологія проєктування навчального об’єкта на основі автоматної моделі		В. О. Остапенко	
	Інформаційна технологія проєктування навчального об’єкта на основі автоматної моделі з додатковою пам’яттю		В. В. Сивоконь	
	Інформаційна технологія проєктування навчального об’єкта на основі вебтехнологій	Ф. Гарсіа-Карбалейра, К. Мартін-Вільяльба, Г. Окрепка, Б. Л. Гарсіа	Ю. Ботузова, Е. Л. Оу, Н. Т. А. Рамаха, К. І. Чонг, А. А. Жако, Б. Марсен, Е. Сімен	А. Де Санто, Б. Дж. Еріксон
	Інформаційна технологія проєктування навчального об’єкта на основі мобільних технологій	Л. К. Джуера	К. Ян, О. І. Кузьмич, Ф. Карузо, С. Барловіц, Е. К. Абад, С. Атін, М. Х. Наранхо Санчес	
	Інформаційна технологія проєктування навчального об’єкта на основі desktop-технологій		К. Ан, О. В. Ольховська, Р. Г. Валєєв, І. В. Перегняк	У. Росміаті, У. Фарук
	Методика створення навчального об’єкта		Н. А. Доценко, В. Н. Хідаят, Т. Капелюшна, В. М. Михалевич, Л. З. Педро, Л. де М.-Ортега	

2 ВИБІР МОДЕЛЕЙ ТА МЕТОДІВ ДОСЛІДЖЕННЯ

2.1 Теорії навчання

Кваліфікаційна робота є кросдисциплінарною, оскільки крім розроблення віртуального тренажера порушується питання педагогіки, а саме — практичне застосування теорій навчання.

До класичних теорій навчання відносять біхевіоризм, когнітивізм та конструктивізм.

В основі біхевіоризму знаходиться ідея формування необхідної поведінки особи, що навчається, під впливом стимулів різного типу. За теорією Скіннера під назвою «operant conditioning», поведінка, що заохочується, матиме вищу ймовірність виконання, а карана — нижчу.

Своєю чергою когнітивізм зосереджується на процесах роботи з інформацією. Підвищення ефективності викладання досягається поліпшенням процесів отримання, організації, зберігання та відтворення навчального матеріалу в мозку.

Навчання із застосуванням конструктивізму полягає у надані матеріалів, що матимуть практичний сенс для учнів/студентів. Відповідно створюється певне середовище з реальною проблемою, яку необхідно вирішити із застосуванням отриманих знань та досвіду взаємодії з нею. У такий спосіб підвищується залученість до навчального процесу.

Конективізм — нова теорія, яка стверджує, що станом на зараз важливим є не обсяг знань людини, а її здатність швидко знаходити, обробляти та використовувати потрібну інформацію. Сучасне навчання — це більше про створення зв’язків із необхідними «базами даних» (включно з людьми — нетворкінг). Причиною таких докорінних змін поглядів є експоненційне зростання кількості інформації.

В основі розроблюваного тренажера знаходяться ідеї одразу кількох теорій навчання.

Когнітивізм застосовується найчастіше. У першу чергу використання табличного представлення транспортної задачі дозволяє користувачу одразу переглянути всі вхідні дані, необхідні для розв'язання. До того ж інформація краще організується в пам'яті людини завдяки чіткому розмежуванню сутностей задачі. Таблиця є інтерактивною. Взаємодія з нею максимально відповідає процесу розв'язання на папері: введення фіктивного учасника перевезень, пошук початкового опорного плану та його оптимізація, оновлення запасів/потреб постачальників/споживачів, видалення їх із розгляду, побудова циклу переходу тощо. Написи на віртуальній дошці утворюють формалізований опис, схожий на той, що пишеться на реальній. Інтерфейс та логіка функціонування вбудованого калькулятора нагадують настільний. Використання візуальних ефектів відбувається для акцентування. Усі розглянуті чинники позитивно впливають на спроможність студентів до зберігання знань у пам'яті та їх відтворення.

Біхевіоризм застосовується під час оцінювання в режимі «Екзамен»: помилки призводять до значних втрат балів. Практичність тренажера є втіленням ідеї конструктивізму. Прослідковується й конективізм: вебзастосунок може слугувати вичерпним джерелом за темою «Транспортні задачі».

2.2 Транспортні задачі

Транспортна задача — спеціальна задача лінійного програмування. Є однією з найбільш поширених. Вона присвячена пошуку оптимальних планів перевезень продукції від постачальників до споживачів.

У рамках кваліфікаційної роботи розглядаємо транспортні задачі в цілих числах з однотипною продукцією. Загальна постановка: нехай у m постачальників $A_1, A_2, \ldots, A_m$ міститься однорідна продукція в кількості $a_1, a_2, \ldots, a_m$ відповідно. Її необхідно перевезти до n споживачів $B_1, B_2, \ldots, B_n$ у кількості $b_1, b_2, \ldots, b_n$ відповідно. Вартість перевезення одиниці вантажу від постачальника A_i до споживача B_j дорівнює c_{ij} (задають матрицю витрат C). Треба знайти план перевезень, щоб виконувалися такі умови:

— уся продукція постачальників вивезена;

— потреби всіх споживачів повністю задоволені;

— загальна вартість перевезень є найменшою [52, с. 130].

Математична постановка задачі [53, с. 150] зводиться до знаходження мінімального значення функції

$$f(X)=\sum_{i=1}^{m}\sum_{j=1}^{n}c_{ij}x_{ij}\to\min \tag{2.1}$$

із дотриманням системи обмежень:

$$\begin{cases} \sum\limits_{j=1}^{n}x_{ij}=a_i, \\ \qquad\qquad \left(i=\overline{1,m};\ j=\overline{1,n}\right) \\ \sum\limits_{i=1}^{m}x_{ij}=b_j, \end{cases} \tag{2.2}$$

та накладанням умов невід’ємності на змінні задачі:

$$x_{ij}\geq 0,\left(i=\overline{1,m};\ j=\overline{1,n}\right) \tag{2.3}$$

Існують різні ускладнення постановки транспортних задач. Прикладами додаткових умов є:

— заборона перевезень між деякими постачальниками та споживачами;

— обов’язкове постачання заданої кількості вантажу від конкретного постачальника до конкретного споживача;

— встановлення мінімальної кількості вантажу від конкретного постачальника до конкретного споживача;

— встановлення максимальної кількості вантажу від конкретного постачальника до конкретного споживача [54].

Усі вони зводяться до звичайної транспортної задачі шляхом перетворень. Ці задачі знаходяться за рамками кваліфікаційної роботи.

План транспортної задачі — це будь-який невід'ємний розв'язок системи рівнянь, утвореної балансовими обмеженнями (2.2). Позначається як матриця $X = (x_{ij})\,(i = \overline{1,m};\ j = \overline{1,n})$, де x_{ij} — кількість продукції, що перевозиться від постачальника A_i до споживача B_j.

Оптимальний план транспортної задачі $X^* = \left(x_{ij}^*\right)(i = \overline{1,m};\ j = \overline{1,n})$ — такий план, за якого цільова функція (2.1) набуває мінімального значення.

Вхідні дані транспортної задачі можна подати в табличному вигляді (див. табл. 2.1).

Таблиця 2.1 — Транспортна таблиця

Спож. / Пост.		B_1	B_2	…	B_j	…	B_n
		b_1	b_2		b_j		b_n
A_1	a_1	c_{11} x_{11}	c_{12} x_{12}	…	c_{1j} x_{1j}	…	c_{1n} x_{1n}
A_2	a_2	c_{21} x_{21}	c_{22} x_{22}	…	c_{2j} x_{2j}	…	c_{2n} x_{2n}
…	…	…	…	…	…	…	…
A_i	a_i	c_{i1} x_{i1}	c_{i2} x_{i2}	…	c_{ij} x_{ij}	…	c_{in} x_{in}
…	…	…	…	…	…	…	…
A_m	a_m	c_{m1} x_{m1}	c_{m2} x_{m2}	…	c_{mj} x_{mj}	…	c_{mn} x_{mn}

Обмеження транспортної задачі в розгорнутому вигляді:

$$\begin{cases} x_{11} + x_{12} + \ldots + x_{1n} = a_1, \\ x_{21} + x_{22} + \ldots + x_{2n} = a_2, \\ \ldots \\ x_{m1} + x_{m2} + \ldots + x_{mn} = a_{mn}, \\ x_{11} + x_{21} + \ldots + x_{m1} = b_1, \\ x_{12} + x_{22} + \ldots + x_{m2} = b_2, \\ \ldots \\ x_{1n} + x_{2n} + \ldots + x_{mn} = b_{mn} \end{cases} \quad (2.4)$$

Задача має $m \times n$ змінних та $m + n$ рівнянь-обмежень.

Також існує її графічний спосіб подання (див. рис. 2.1). Кожен відрізок $A_i B_j$ є комунікацією — шляхом постачання продукції обсягом x_{ij} від постачальника A_i до споживача B_j.

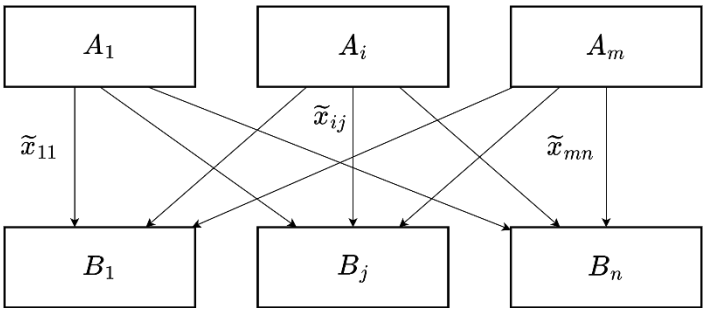


Рисунок 2.1 — Графічний спосіб подання транспортної задачі

Коефіцієнти при невідомих системи (2.4) утворюють матрицю, вектори-стовпці якої називаються векторами комунікацій і позначаються як P_{ij}:

$$P_{ij} = \left(\underbrace{0,\ldots,0,\underbrace{1}_{i},0,\ldots,0,1,0,\ldots,0}_{m+j} \right)^T. \quad (2.5)$$

Вільні члени утворюють вектор запасів-потреб P_0, де:

$$P_0 = \left(a_1, a_2, \ldots, a_m, b_1, b_2, \ldots, b_n\right)^T. \quad (2.6)$$

У такий спосіб систему обмежень можна записати у вигляді:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} P_{ij} = P_0. \quad (2.7)$$

План транспортної задачі вважається опорним, якщо вектори комунікацій P_{ij}, які відповідають додатним елементам x_{ij} плану, є лінійно незалежними [52, с. 132].

Існує теорема [55, с. 7], що ранг матриці коефіцієнтів системи обмежень (2.4) дорівнює $m+n-1$. Відповідно плани перевезень поділяють на невироджені та вироджені. До перших належать ті, що містять $M = m+n-1$ додатних елементів x_{ij}, а до других — у яких $M < m+n-1$ [52, с. 132].

Базис транспортної задачі — довільна система $m+n-1$ лінійно-незалежних векторів P_{ij}, яка включає ті, що відповідають додатнім перевезенням плану.

Транспортні задачі бувають закритими та відкритими, залежно від початкового співвідношення між загальними запасами постачальників та загальними потребами споживачів.

Необхідна і достатня умова існування розв'язку (умова балансу) має вигляд:

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j. \quad (2.8)$$

Транспортна задача, для якої вона виконується, називається закритою (збалансованою), а для якої не виконується — відкритою (незбалансованою).

Для відкритих транспортних задач виконується лише частина балансових обмежень (2.2). Якщо загальні запаси постачальників перевищують загальні потреби споживачів, тобто $\sum_{i=1}^{m} a_i > \sum_{j=1}^{n} b_j$, тоді система обмежень має вигляд:

$$\begin{cases} \sum_{j=1}^{n} x_{ij} \le a_i, \\ \qquad \left(i = \overline{1,m};\ j = \overline{1,n}\right). \\ \sum_{i=1}^{m} x_{ij} = b_j, \end{cases} \tag{2.9}$$

Для розв'язання треба ввести фіктивного споживача B_{n+1} із потребами $b_{n+1} = \sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j$ та встановити тарифи $c_{i\,n+1} = 0, \left(i = \overline{1,m}\right)$.

Аналогічно, якщо загальні потреби споживачів перевищують загальні запаси постачальників, тобто $\sum_{j=1}^{n} b_j > \sum_{i=1}^{m} a_i$, тоді система обмежень має вигляд:

$$\begin{cases} \sum_{j=1}^{n} x_{ij} = a_i, \\ \qquad \left(i = \overline{1,m};\ j = \overline{1,n}\right). \\ \sum_{i=1}^{m} x_{ij} \le b_j, \end{cases} \tag{2.10}$$

Для розв'язання треба ввести фіктивного постачальника A_{m+1} із запасами $a_{m+1} = \sum_{j=1}^{n} b_j - \sum_{i=1}^{m} a_i$ та встановити тарифи $c_{m+1\,j} = 0, \left(j = \overline{1,n}\right)$.

Після перетворень балансові обмеження (2.2) починають виконуватися. Перевезення між реальними учасниками перевезень та фіктивним не впливають на значення цільової функції: тарифи дорівнюють 0.

Симплекс-метод застосовний для розв'язання транспортної задачі, утім особливості системи обмежень (2.4) (умова задаються рівняннями, кожне

невідоме зустрічається лише два рази, одиничні коефіцієнти цільової функції при невідомих) дозволяють використовувати його спрощений варіант.

Процес розв'язання транспортної задачі складається з трьох етапів [52, с. 133]:

1. знаходження початкового опорного плану;
2. дослідження отриманого плану на оптимальність;
3. перехід від одного опорного плану до іншого.

Виділяють дві групи методів для отримання опорного плану транспортної задачі: прямі та ті, що потребують додаткових обчислень. До першої належать метод північно-західного кута, метод мінімальної вартості та їхні модифікації. До другої — метод Фогеля, метод Лебедєва тощо.

Перевагою прямих методів є простота застосування. Вони дозволяють швидко знайти початковий опорний план, утім він часто є далеким від оптимального, що в подальшому вимагатиме більшої кількості ітерацій оптимізації. Водночас методи другої групи дозволяють отримати кращий початковий розв'язок, але це потребуватиме значних обчислень.

У рамках кваліфікаційної роботи розглядаємо лише метод північно-західного кута та метод мінімальної вартості, оскільки на них найбільше зосереджуються викладачі дисципліни «Методи дослідження операцій» під час викладення теми «Транспортні задачі».

Застосування цих методів базується на табличному представленні задачі (див. табл. 2.1). Клітини з додатними перевезеннями називаються зайнятими, їм відповідають базисні змінні опорного плану. Інші клітини — незайнятими (нульові перевезення в таблиці позначаються порожніми місцями), їм відповідають вільні змінні [52, с. 134].

За методом північно-західного кута на кожному кроці заповнюється доступна ліва верхня клітина. Побудова опорного плану починається з A_1B_1. Метод є простим, але результат часто далекий від оптимального через інваріантність до значень тарифів.

Натомість метод мінімальної вартості їх враховує, що дозволяє отримати кращий опорний план. При цьому не вимагається значних зусиль для застосування: на кожному кроці треба заповнювати серед доступних клітину з найменшим тарифом. Важливо: клітини з нульовими тарифами, що відповідають фіктивному учаснику перевезень, повинні розглядатися в останню чергу.

Під час пошуку початкового опорного плану варто слідкувати за дотриманням теореми про ранг, що задає необхідну кількість базисних клітин, яка дорівнює $m+n-1$. Якщо на якомусь кроці, що не є останнім, за результатами постачання запаси постачальника та потреби споживача одночасно стають рівними 0, тоді необхідно обрати одного з них для усунення з розгляду, а для іншого, перш ніж також прибрати, визначити базисну клітину з поставкою 0. У такому разі план стає виродженим.

Під час заповнення клітини A_iB_j обсяг постачання обчислюється за формулою: $x_{ij}=\min\{a_i,b_j\}$, де a_i — поточні запаси постачальника A_i, b_j — поточні потреби споживача B_j.

Метод потенціалів дозволяє легко дослідити попередньо отриманий опорний план на оптимальність, а також пропонує механізм переходу до наступного не гіршого розв'язку — шляхом застосування циклу переходу.

Потенціали постачальників та споживачів — числа, що позначаються як $u_i(i=\overline{1,m})$ та $v_j(j=\overline{1,n})$ відповідно.

Залежність значень потенціалів постачальників та споживачів описується формулою:

$$c_{ij}=u_i+v_j, \tag{2.11}$$

де c_{ij} — тариф заповненої клітини A_iB_j,

u_i — потенціал постачальника A_i,

v_j — потенціал споживача B_j.

Кількість потенціалів відповідає кількості учасників перевезень: $n+m$, а кількість невідомих у системі лінійних рівнянь (2.4) дорівнює $n+m-1$. У такий спосіб отримуємо систему потенціалів із $n+m-1$ рівняннями та $n+m$ змінними. У такій постановці вона має безліч розв'язків, а тому для отримання одного з них один із потенціалів можна ініціалізувати довільним значенням. Зазвичай для зручності ним виступає 0.

Для дослідження оптимальності розв'язку використовують характеристики вільних клітин — значень E_{ij}, які обчислюються для всіх вільних клітин A_iB_j, за формулою: $E_{ij} = c_{ij} - (u_i + v_j)$.

Критерій оптимальності розв'язку транспортної задачі: невід'ємні характеристики всіх вільних клітин. Наявність серед них 0 свідчить про існування альтернативного оптимуму.

Характеристика вільної клітини відображає зміну значення цільової функції (2.1) в разі перерозподілу до неї одиниці продукції із застосуванням циклу переходу. Від'ємне значення покращує розв'язок, оскільки:

$$f(X^{(k+1)}) = f(X^{(k)}) + E_{ij}x_{ij}, \qquad (2.12)$$

де k — номер попереднього плану,

$X^{(k)}$ — попередній план,

$X^{(k+1)}$ — наступний план,

E_{ij} — характеристика вільної клітини до якої відбувається перерозподіл,

x_{ij} — обсяг поставки (відповідає обсягу перерозподілу Δ).

Цикл переходу — замкнена ламана, що описує спосіб зміни обсягів поставок у такий спосіб, щоб перейти до наступного плану транспортної задачі без порушення обмежень (2.2) за рядками (постачальниками) та стовпчиками (споживачами).

Його побудова починається з вільної клітини з найменшою оцінкою. Далі цикл проходить через базисні клітини, кожен раз змінюючи напрямок на 90°. Кожній вершині приписується знак: «+» — для непарних і «−» — для парних [54]. Позначки відображають, із яким знаком додаватиметься обсяг перерозподілу Δ, який визначається як мінімальне значення серед базисних клітин циклу під знаком «−». Побудова завершується в місці її початку. Важливо: для будь-якої вільної клітини існує тільки один цикл переходу.

Крім методу потенціалів існує дуже схожий на нього — розподільчий метод із аналогічними можливостями: перевірка оптимальності розв’язку та оптимізація. Замість оцінок вільних клітин обчислюються покращення цільової функції від виконання перерозподілу до кожної з них, після чого виконується перехід за найрезультативнішим циклом переходу. Хоч і економимо час на визначенні потенціалів, на побудову всіх циклів переходу для знаходження обсягів перерозподілу, а також обчислення забезпечуваних покращень витрачається значно більше. Метод потенціалів дозволяє уникнути таких складних дій, тому він популярніший. Відповідно його обрано і для програмної реалізації.

2.3 Автоматна модель віртуального тренажера

За результатами аналізу предметної області з’ясовано, що тренажери створюються для такого кола задач, процес розв’язання яких можна подати у вигляді скінченної кількості кроків. Інакше кажучи, для яких існують алгоритми розв’язання.

Логіку більшості тренажерів можна змоделювати за допомогою скінченного автомата — математичної абстракції пристрою, що в будь-який момент часу знаходиться в одному зі скінченної множини стані. Керувальний вплив надходить у вигляді символів вхідного алфавіту, що призводять до переходів між станами та повернення символів із алфавіту виходів.

Такий підхід використаний у роботі «Методичні положення з побудови комп’ютерних віртуальних тренажерів у системах дистанційного навчання», де

В. О. Остапенко застосував елементи теорії автоматів, щоб провести математичне моделювання навчального об’єкта. На думку автора роботи віртуальний тренажер — це система, що складається з множини станів, множини керувальних впливів, множини правил, множини функціональних блоків та множини помилкових станів. Перша складається зі станів, що визначаються в просторі параметрів тренажера. Множиною керувальних впливів є всі можливі дії користувача. Правила — трійка (стан, керувальний вплив, наступний стан), за допомогою якої описуємо логіку переходів між станами. Функціональні блоки складаються з множини правил та множини цільових правил. Автор пропонує їх використовувати для декомпозиції логіки всього тренажера: для блоків визначаються, які правила діють, та якого стану-цілі треба досягти. Множина помилкових станів застосовувалася для позначення станів непрацездатності тренажера, що вже є специфікою, оскільки розроблялася модель тренажера генератора імпульсів точної амплітуди [50, с. 118].

Ці напрацювання узагальнено для отримання формалізованого опису автоматної моделі віртуального тренажера. Нехай $A=(S,X,Y,\delta,\lambda,s_0,F)$ — детермінований скінченний ініціальний автомат, що описує деякий віртуальний тренажер, де:

— S — скінченна множина станів, що являють собою завдання тренажера та спеціальні стани;

— X — скінченний вхідний алфавіт, елементами якого є всі можливі дії та відповіді користувача;

— Y — скінченний вихідний алфавіт, що складається з можливих повідомлень-реакцій тренажера на дії та відповіді користувача на завдання;

— $\delta: S\times X\to S$ — функція переходів, яка визначає правила переходів між завданнями та спеціальними станами тренажера;

— $\lambda: S\times X\to Y$ — функція виходів, що задає повідомлення-реакції тренажера на основі поточного завдання або спеціального стану та надаваної відповіді або дії;

— s_0 — початковий стан ($s_0 \in S$), є спеціальним станом тренажера;

— F — множина кінцевих станів ($F \subset S$), які є спеціальними станами тренажера.

Початковий стан s_0 потрібний для проведення ініціалізації тренажера. Множина завдань $\{s_i | i = \overline{1,n}\} \subset S$, де n — їхня кількість. Досягнення одного з кінцевих станів множини F свідчить про завершення роботи над тренажером.

Вхідний алфавіт X — скінченна множина. Це справедливо для завдань як закритого типу — за визначенням, так і для відкритого — у них на власні відповіді користувача накладаються різні типи обмежень, одним із яких є максимальна кількість символів. Воно може встановлюватися явно або неявно витікати з такого технічного параметру пристрою як доступна пам'ять.

Функція переходів δ є відображенням, яке кожній упорядкованій парі елементів (s, x), де $s \in S$, а $x \in X$, ставить у відповідність елемент із множини S. Створювана множина правил $\{(s_i, x_k, s_j) | i = \overline{1,n},\ j \in [1,n], k = \overline{1,m}\}$, де $m = | X |$ — кількість варіантів дій та відповідей, визначає логіку функціонування тренажера: наслідки дій користувача (теоретично можливою є реалізація завершення виконання тренажера на довільному етапі, повернення до попереднього кроку або повний перезапуск), а також правильні та неправильні відповіді для кожного завдання.

Функція виходів λ є відображенням, яке кожній упорядкованій парі елементів (s, x), де $s \in S$, а $x \in X$, ставить у відповідність елемент із множини Y. Створювана множина правил $\{(s_i, x_k, y_l) | i = \overline{1,n}, k = \overline{1,m}, l \in [1, N]\}$, де N — кількість варіантів повідомлень тренажера, дозволяє охарактеризувати події, що відбуваються під час функціонування тренажера: «правильна відповідь», «неправильна відповідь» та інше.

Множина кінцевих станів F є власною підмножиною S, оскільки припускаємо, що множина завдань тренажера не є порожньою: $\{s_i | i = \overline{1,n}\} \neq \varnothing$.

Такт автомата t — значення лічильника виконаних кроків, що позначає абстрактний момент часу, у якому автомат знаходиться після прийняття t вхідних символів (у нашому випадку — відповідей на завдання тренажера). Початкове значення дорівнює 0. Крок дискретизації відліку часу $\Delta t = 1$, тобто: $t \to t + \Delta t$.

Наступне завдання тренажера $s(t+1)$ у момент часу t визначається за формулою: $s(t+1) = \delta(s(t), x(t))$, де $s(t)$ — поточне завдання, $x(t)$ — прийнята відповідь на завдання.

Реакція тренажера $y(t)$ у момент часу t визначається за формулою: $y(t) = \lambda(s(t), x(t))$, де $s(t)$ — завдання та $x(t)$ — відповідь на нього.

За допомогою описаного кінцевого автомата можна проводити моделювання різних тренажерів, утім далеко не всіх. Серед вимог до цих програмних продуктів зазвичай виділяють підтримку достатньої варіативності завдань, яку можна забезпечити шляхом використання різних методів генерації псевдовипадкових чисел. Ця проблема частково вирішується в межах звичайної моделі шляхом введення паралельних гілок станів. Принцип роботи: функціонування тренажера розпочинається зі стану s_0 — подаємо на вхід випадкове значення, згенероване окремою підсистемою, після чого на його основі буде обраний варіант у вигляді гілки завдань. Для більш гнучкої варіативності можна створювати локальні розгалуження.

Інша досить поширена проблема — необхідність використання інформації, отриманої раніше. У якості прикладу розглянемо покрокове виконання постачання в обрану клітину тарифу в транспортній задачі. Нехай процес розв’язання не є транзакційним — зміни застосовуються поступово, а тому під час окремих тактів автомата задача може знаходиться в неузгодженому стані: спочатку користувач має вказати обсяг поставки, після чого оновити дані про запаси постачальника та потреби споживача. Це три окремі послідовні завдання. Відповідь на перше впливатиме на відповіді на подальші завдання, а, відповідно, на переходи.

Теоретично, проблема вирішувана за допомогою підвищення інформаційної спроможності станів, що передбачає збільшення розмірності простору визначення параметрів, у якому визначаються стани. Стан — це набір значень параметрів у певний момент часу (такт автомата). За своєю природою є статичним (незмінним), оскільки будь-яка зміна означала б перехід до іншого. Запам’ятовування попередньої інформації можна реалізувати шляхом її кодування за допомогою станів. Варіативність значень змінних, які треба зберегти, впливає на складність автоматної моделі: більша варіативність — більше станів.

Серед скінченних автоматів за критерієм пам’яті можна виділити ті, які працюють без неї, та ті, що її використовують. Перші з огляду на перелічені вище недоліки не підходять для математичного моделювання віртуальних тренажерів зі складною логікою переходів між завданнями.

Своєю чергою другі дозволяють це зробити. Серед наявних моделей найбільш відомими є автомат із магазинною пам’яттю (pushdown automaton), автомат із двома стеками (two stack pushdown automaton), автомат на основі черги (queue automaton) та автомат із вкладеними стеками (nested stack automaton). Вони розроблялися для моделювання процесів різної складності та дослідження питання обчислюваності. Варто відмітити, що кожна з моделей накладає обмеження на доступ до пам’яті.

Очевидно, що питання обчислюваності для тренажерів вирішене: можна розробити алгоритм, який виводить завдання та очікує на відповідь користувача. Якщо та правильна — перехід до наступного завдання, якщо ж ні — відображення повідомлення про помилку та очікування виправлень (типова логіка обробки цієї ситуації). Може потребуватися використання змінних, щоб використовувати інформацію з попередніх завдань.

Автоматна модель тренажера — це інструмент проєктування логіки його роботи. Ключовою перевагою формалізованого опису, побудованого на математичному апараті теорії автоматів, над змістовним є чіткість. У такий спосіб усувається суб’єктивна складова. Підхід стає особливо вдалим, якщо розробляється

тренажер математичного спрямування. Деталізований опис завдань-станів та переходів пришвидшує процес розроблення програмної реалізації.

Наявна автоматна модель віртуального тренажера не може бути використана для опису логіку переходів між завданнями тренажера, присвяченого покроковому розв'язанню транспортних задач. У нашому випадку виникає потреба в доступі до попередніх вхідних даних та функції переходів, що використовувала б значення з пам'яті.

Автоматну модель віртуального тренажера можна розширити шляхом введення механізму роботи з пам'яттю, причому з огляду на рівень абстракції, на якому інструмент проєктування застосовуватиметься, вважатимемо, що на доступ до пам'яті обмеження не накладаються (матимемо випадковий доступ до пам'яті — RAM). Тоді формалізований опис матиме вигляд: нехай деякий віртуальний тренажер описується за допомогою $A=(S,X,Y,\delta,\lambda,s_0,F,M,\mu,m_0)$ — детермінований скінченний ініціальний автомат із додатковою пам'яттю з довільним доступом, що описує деякий віртуальний тренажер. Ключові відмінності від базової моделі:

— скінченна множина можливих станів пам'яті M, елементами якої є всі можливі комбінації значень параметрів у просторі визначення станів пам'яті, що можуть зустрітися під час роботи тренажера;
— на початку роботи (до прийняття в стані s_0 вхідного символу з множини X) пам'ять вважається порожньою — уміст такого стану m_0 позначається як ε;
— функція переходів враховує стан пам'яті $\delta: S\times X\times M\to S$;
— аналогічно для функції виходів $\lambda: S\times X\times M\to Y$;
— функція зміни станів пам'яті $\mu: S\times X\times M\to M$;
— наступний уміст пам'яті в моменту часу t визначається за формулою $m(t+1)=\mu(s(t),x(t),m(t))$.

Кожен елемент множини M являє собою кортеж — точку в багатовимірному просторі визначення параметрів станів пам'яті. Компоненти можуть бути різного типу залежно від особливостей логіки тренажера, що моделюється.

Робота за розширеною автоматною моделлю в момент часу t відбувається на основі такого алгоритму:

1. виконання логіки, пов'язаної з поточним станом $s(t)$;
2. очікування на елемент $x(t)$ із вхідного алфавіту X;
3. пошук значення $s(t+1)$ функції переходів δ;
4. пошук значення $y(t)$ функції виходів λ;
5. пошук значення $m(t+1)$ функції зміни станів пам'яті μ;
6. повернення результату роботи у вигляді $y(t)$;
7. перехід до наступного стану пам'яті $m(t+1)$;
8. перехід до наступного стану тренажера $s(t+1)$.

Отримана модель дозволяє зберігати концепцію завдань у станах: як формується питання, яка очікувана відповідь. Можливість звернення до пам'яті та високий рівень абстрагування дозволяють у процесі проєктування перевикористовувати стани, завдяки чому можна побудувати автоматну модель простої структури, для якої діаграма станів вийде наочною.

Виділяють два основні підходи до побудови діаграм: декларативний та імперативний. Перший дозволяє зосередитися на описі бажаного результату без вказання конкретних дій для його досягнення. Другий, навпаки, передбачає безпосереднє керування процесом.

Існує велика кількість інструментів, що дозволяють будувати діаграми. У першу чергу розглянуті програмні продукти, що працюють за декларативним підходом, оскільки він є більш бажаним, оскільки, теоретично, він здатен значно спростити роботу завдяки автоматизації.

Graphviz — інструмент, що дозволяє створювати діаграми графів на основі декларативного підходу. Для вказання потрібного результату використовується

DOT — мова опису графів. Для автоматичного розташування їхніх вузлів пропонується декілька алгоритмів, серед яких dot, neato, fdp та інші. Надаються широкі можливості оформлення елементів діаграми: колір та заливка, розмір та форма вузлів, стилі з'єднань та властивості тексту. Інструмент здатен будувати візуалізацію графів різного типу, як орієнтованих, так і неорієнтованих. Додатково є підтримка підграфів та структур.

Graphviz має й суттєві недоліки. У першу чергу, вивчення синтаксису DOT потребує часу. Автоматичне розташування працює, утім часто не відповідає вимогам, особливо в разі побудови графів із великою кількістю вузлів. Серед відомих проблем: перекриття вузлів, перетин ребер та неочікуване розташування елементів — усе це погіршує сприйняття діаграми. Імперативний підхід хоча й підтримується в разі використання алгоритму fdp, утім задавати положення вузлів за координатами надзвичайно складно. Виведення LaTeX-формул не підтримується.

Існують програмні продукти для роботи з LaTeX, такі як TeXWorks, TeXstudio, Texmaker. Крім створення професійних текстових документів вони можуть бути використані для побудови діаграм із формулами завдяки використанню пакетів, як-от Tikz та dot2texi.

Вони також мають і недоліки. По-перше, час встановлення дистрибутиву MiKTeX та самого IDE для роботи з LaTeX-розміткою. По-друге, Tikz дозволяє створювати якісні діаграми, утім за імперативним підходом, що потребує великої кількості коду для ручного розташування елементів. По-третє, є складнощі з керуванням залежностями: для підключення до віддалених репозиторіїв із необхідними пакетами потрібен VPN. Якщо використовувати dot2texi — матимемо недоліки Graphviz, крім підтримки LaTeX, і потребу в додаткових налаштуваннях, таких як shell-escape (для можливості виконання зовнішніх команд під час компіляції) та інші.

Overleaf — вебсервіс, що дозволяє створювати документи за основі LaTeX-розмітки. Він вирішує недоліки desktop-аналогів. Сервіс пропонує вже

налаштоване середовище, що економить час. У Overleaf немає потреби в ручному керуванні залежностями — можна одразу використовувати потрібні пакети. Він доступний будь-де та будь-коли за наявності Інтернет-з'єднання, може використовуватися в якості сховища для проєктів на LaTeX. Сайт дозволяє працювати в команді: можна надавати спільний доступ для читання та редагування. Для спрощення роботи пропонує велику кількість шаблонів.

Водночас Overleaf має й недоліки. Швидкодія вебсервісу є нижчою в порівнянні з локальною машиною. До того ж має суттєві тарифні обмеження на рівень швидкодії, максимальний час компіляції, кількість учасників на проєкт та інші. До того ж безкоштовний план не передбачає ведення історії змін у документах. Можуть виникати складнощі в разі виникнення необхідності у використанні специфічних пакетів. Крім того раніше сформований документ колись може перестати компілюватися, якщо використовувана залежність буде оновлена на серверах і виявиться несумісною.

Отже, можливості інструментів, що працюють за декларативним підходом, виявилися недостатніми для створення діаграм тренажера з транспортних задач. Через це довелося звернутися до імперативного підходу. Застосунки, що працюють на його основі, досить часто реалізують також WYSIWYG — підхід, що передбачає можливість перегляду результату під час редагування.

Adobe InDesign — потужне середовище для створення професійних друкованих документів. Також використовується для розроблення інфографіки, зокрема діаграм. Програмний продукт втілює імперативний підхід у максимальному прояві: можливості обмежуються лише креативністю автора. Застосунок має потужну підтримку стилів для об'єктів, абзаців, символів, таблиць та їхніх комірок, що дозволяє легко підтримувати консистентність оформлення елементів. Надає незначні можливості автоматизації, як-от автоматичне визначення довжини/ширини за контентом та дотримання внутрішніх відступів для текстових блоків.

Adobe InDesign має свої недоліки. Одним із них є вартість ліцензії на продукт. Інтерфейс програми навантажений великою кількістю інструментів та

налаштувань — його складність може сповільнювати роботу. Безпосередня підтримка введення математичних виразів у професійному вигляді відсутня, що частково вирішується за допомогою сторонніх скриптів і плагінів або вебсервісів для створення зображень для LaTeX-формул. Система прив'язок незручна для розташування стрілок: можна здійснити прив'язку лише до одного об'єкта; до того ж відбуватиметься переміщення разом із головним об'єктом, а не розтягування — стрілки доводиться налаштовувати вручну, що вимагає значних часових витрат під час роботи з великими діаграмами.

Figma — популярний інструмент для розроблення дизайну з потужною підтримкою спільної роботи. Можна працювати як із desktop, так і з вебверсією. Проєкти за замовчуванням зберігаються в хмарному сховищі. Для підтримки одноманітності оформлення елементів можна визначати компоненти та варіанти, палітри кольорів та шрифтів. Механізм обмежень (constraints) дозволяє задавати правила розташування об'єктів відносно один одного — у такий спосіб визначається реакція на розтягування та стискання елементів. Автоматичне розташування (auto layout) дозволяє зекономити час: елементи контейнера позиціонуватимуться автоматично (не треба переставляти вручну); підтримується динамічний контент за розмірами та за кількістю елементів; застосовуються сталі відступи.

Серед недоліків Figma є робота зі стрілками: вони не прив'язуються до об'єктів. Проблема частково вирішувана за допомогою стороннього плагіну Arrow Auto: позиція з'єднання оновлюється, утім положення не змінюється, у результаті чого може утворюватися перетин. До того ж не можемо обрати сторони, які з'єднуються.

Figma не підтримує LaTeX-формули, що можна дещо компенсувати плагіном Latex Editor, який дозволяє генерувати SVG-зображення формул та одразу їх вбудовувати в документ. На жаль, редагувати результат напряму неможливо.

Miro — вебсервіс, що використовується для створення інфографіки. Дозволяє створювати між блоками адаптивні стрілки, які автоматично

підлаштовуються під оновлене розташування пов’язаних об’єктів без перетину вузлів ребрами. Також можна обирати, до яких точок прив’язуватися. Підтримується командна робота над створенням діаграм.

Варіант не позбавлений недоліків. Вбудована підтримка LaTeX-формул відсутня — треба завантажувати плагін, наприклад, MiroTeX, що дозволяє на основі введеної формули генерувати векторні зображення. У такий спосіб проводити редагування раніше створених формул неможливо — треба створювати новий об’єкт заново та видаляти попередній. Середовище не дозволяє налаштовувати внутрішні відступи (padding) для фігур. Подолання цього обмеження потребує створення додаткових елементів, що не дуже зручно. Підтримка консистентності ускладнена: зовнішній вигляд кожної фігури налаштовується окремо, оскільки централізоване керування стилями відсутнє. У безкоштовній версії значною мірою обмежена якість діаграм, що експортуються.

Draw.io — безкоштовний кросплатформовий додаток на базі Electron для створення діаграм. Також доступний із браузера. Пропонує до використання велику кількість фігур та типів діаграм. Підтримує відображення LaTeX-розмітки без необхідності встановлення додаткових плагінів. Представлена вбудована можливість нумерації елементів діаграми. Стрілки прив’язуються до з’єднуваних об’єктів і адаптуються під час переміщень. Підтримує різні формати експорту, серед яких PNG, JPEG, SVG та багато інших, завдяки чому в результаті роботи можна отримати діаграми високої якості. Зручна сітка допомагає в розташуванні елементів. Є механізм спільної роботи.

Недоліків у draw.io небагато. Інколи у результаті перетягування голівки стрілок можуть неправильно обертатися, що потребує ручного корегування. Можливості оформлення елементів дещо обмежені: не можна створювати власні стилі.

Для побудови діаграм завдань тренажера з транспортних задач обраний draw.io, оскільки він є найбільш ефективним для виконання цього завдання. Фактори, що вплинули на вибір: зручна робота зі стрілками, вбудована підтримка

LaTeX-формул із можливістю легкого оновлення, легкість забезпечення консистентності оформлення та висока якість вихідних файлів.

Під час створення діаграм (див. додатки А.1, А.2, А.3) використані умовні позначення.

Чорне коло позначає ініціальний стан s_0, що відповідає за ініціалізацію тренажера. Серед підпроцесів: обрання режиму роботи, створення умови задачі та змінення налаштувань, зокрема локалізації. Вони не наводяться на діаграмі через рівень абстрагування. Після завершення підготовчих робіт ініціюється перехід до завдань за внутрішнім сигналом «старт».

Чорний круг — кінцевий стан автомата. Використання додаткової пам'яті підвищує здатність діаграми до узагальнення, завдяки чому $|F|=1$.

Стани-завдання тренажера зображуються у вигляді прямокутників, що вказує на виконання додаткової логіки, зокрема формування тексту запитання та проведення маніпуляцій над пам'яттю. Для написання якісного текстового супроводу із точними посиланнями на елементи застосовується нумерація.

Зазвичай на діаграмах станів кінцевих автоматів біля стрілок вказуються вхідні символи, потрібні для здійснення переходів. Простота позначення обумовлена тим, що δ — функція однієї змінної (від вхідного символу x). Своєю чергою запропонована вдосконалена автоматна модель віртуального тренажера може працювати з пам'яттю, що дозволяє задавати більш складні умови для переходів між станами. Щоб розрізнити вимоги до вхідного символу від тих, що стосуються значень у пам'яті, використовується більш складна нотація: за допомогою виразу $x(t)$ посилаємося на вхідний символ, коли автомат знаходиться в стані, звідки виходить стрілка на діаграмі. Зазначається аргумент t, бо неможливо заздалегідь передбачити, у який момент часу відбуватиметься ця перевірка. Таке умовне позначення спрощує процес моделювання.

Для опису складних умов переходів використовуються логічні оператори, зокрема кон'юнкції «$\wedge$» та диз'юнкції «$\vee$», а також дужки — для керування порядком операцій та підвищення зрозумілості виразів. Квантифікатори застосовні

для формалізації кількісних вимог. Під час створення діаграм (див. додатки А.1, А.2, А.3) знадобилися квантифікатори існування «$\exists$», неіснування «$\nexists$», єдиності «$\exists!$» та загальності «$\forall$». Якщо висуваються різні вимоги до значення $x(t)$ у залежності від інших умов, тоді використовується фігурна дужка. Так умову переходу можна записати більш компактно, аніж із використанням логічних операторів та дужок.

Вводяться додаткові символи для позначення ділянок пам'яті, що зберігають проміжні дані. Серед вимог до іменування:

— назва має складатися з одного символу (дозволяється використання нижнього та верхнього індексів);
— відсутність збігів із загальновідомими позначеннями (наприклад, математичними константами: π, e) та тими, що використовуються в предметній області задачі, процес розв'язання якої моделюється;
— недопущення перевикористання — кожна змінна повинна визначатися для однієї цілі, щоб уникнути плутанини.

Для візуального виокремлення додаткових змінних пропонується використовувати діакритичний знак — надрядкову тильду. У якості прикладу: $\tilde{M}$ — обраний користувачем метод знаходження опорного плану перевезень.

Під час іменування варто враховувати семантику. Це сприятиме утворенню асоціативних зв'язків, що пришвидшуватиме інтерпретацію діаграми. Наприклад, змінні $\tilde{s}, \tilde{c}$ — номери постачальника (англ. supplier) та споживача (англ. consumer) відповідно.

Символ «:=» використовується для опису дій над пам'яттю. Він вживається під час визначення додаткових змінних. Прикладом є завдання 6, де обчислюється різниця між загальними запасами та загальними потребами, після чого результат зберігається в $\tilde{d}$. Також застосовний для оновлень значень. Приклад: актуалізація запасів постачальника та потреб споживача після здійснення постачання.

Однотипні стани-завдання дозволено групувати, щоб уникнути створення великої кількості схожих між собою стрілок із підписами. Прикладом є завдання 12, 13 та 14, що присвячені обранню базисної клітини за методом, назва якого зберігається під символом $\tilde{M}$. Перехід до завдання 15 показаний у вигляді однієї стрілки з комплексною умовою. Подання різних перевірок у вигляді альтернатив за значенням $\tilde{M}$ є простішим для сприйняття, аніж три стани з відокремленими описами логіки обробки відповідей користувача. До того ж повернення до обрання базисної клітини в подальшому може відбуватися після надання правильної відповіді на завдання 17, 20 або 21, що потребувало б наведення трьох описів переходів для кожного з них — ще 9 стрілок. Завдяки групуванню діаграму вдалося спростити: маємо 4 переходи замість 12. Варто зазначити, що заміна групи завдань одним узагальненим під час розроблення програмної реалізація не проводиться з міркувань підтримки розширюваності.

Інколи може виникати необхідність у додаткових діях у разі виконання переходу. Як приклад: збереження обраного методу побудови опорного плану перевезень. Перш ніж зберегти значення до пам’яті необхідно впевнитися в його правильності: можливою я ситуація, коли користувач не обрав жодний серед запропонованих, що формально записується як $x(t) = \varnothing$. Рівень абстракції не дозволяє виокремити цю логіку в окремий стан, оскільки це б означало створення додаткового завдання. Також ці дії не можуть бути віднесені до наступного завдання, бо там відбувається зациклювання: після завершення роботи над попередньою базисною клітиною в рамках завдань 17, 20 або 21 відбувається перехід до обрання наступної за заздалегідь обраним методом (група завдань 12–14). До того ж запис виду $\tilde{M} = x(t-1)$ був би помилковим, бо тип даних у $x(t-1)$ залежатиме від того, який був попередній стан. Повертатися до завдання 11 у рамках циклу також не має сенсу, бо процедура знаходження опорного плану повинна бути однорідною — за однаковим методом. Більш того це створюватиме незручності для користувача у вигляді постійної необхідності у вказанні методу побудови опорного плану. Вдалим рішенням є позначення на стрілках додаткової

логіки одразу після умови переходу. Роздільник «|» використовується для поліпшення сприйняття.

Щоб не погіршувати зрозумілість діаграми дозволяється не вказувати переходи до поточного завдання в разі надання неправильних відповідей, зміни локалізації, перезапуску або завершення тренажера в будь-який момент — це властивості будь-якого завдання. Достатньо лише примітки про це, якщо поведінка стандартна та повністю підтримується технологією. В іншому випадку — особливості мають бути відображені.

Аналогічно з результатами функції виходів λ: вважаємо, що будь-який перехід від одного завдання до іншого повертатиме повідомлення про подію типу «правильна відповідь», а якщо до того ж самого — «неправильна відповідь» (за замовчуванням). Перехід від ініціального стану до першого завдання — «початок виконання тренажера», а якщо до кінцевого — «завершення виконання тренажера». Повернення до першого завдання в рамках перезапуску — «перезапуск тренажера». Якщо відбувається зміна налаштувань — «зміна налаштувань». Від тренажера не вимагається виведення таких текстових повідомлень, утім обробка відповідних подій — обов’язкова.

Спроєктовано 34 завдання для тренажера на тему «Транспортні задачі». Вони відображені на діаграмах (див. додатки А.1, А.2, А.3). Завдання утворюють три основні етапи:

— визначення основних властивостей транспортної задачі, приведення до закритого типу;
— знаходження опорного плану перевезень;
— застосування методу потенціалів.

Усе починається з введення кількості постачальників m (завдання 1) та споживачів n (завдання 2). Для надання правильної відповіді студент повинен звернути увагу на кількість рядків та стовпчиків у транспортній таблиці, що містять інформацію про учасників перевезень. Після цього треба обчислити загальні

запаси постачальників $\sum_{i=1}^{m} a_i$ та загальні потреби споживачів $\sum_{j=1}^{n} b_j$ у рамках виконання завдань 3 та 4 відповідно.

Після переходу до завдання 5 відбувається обчислення різниці між загальними запасами постачальників та загальними потребами споживачів. Результати зберігаються в змінній $\tilde{d}$. Користувач повинен дати відповідь на запитання: яке співвідношення між величинами $\sum_{i=1}^{m} a_i$ та $\sum_{j=1}^{n} b_j$. Варіанти відповідей: більше «>», менше «<» та дорівнює «=». Інакше кажучи, який знак у значення $\tilde{d}$, що можна з'ясувати за допомогою кусково-заданої функції sgn. У такий спосіб:

— 1 — запаси більше потреб;

— 0 — запаси дорівнюють потребам;

— -1 — запаси менше потреб.

Завдання 6 присвячено визначенню типу транспортної задачі. Вони бувають відкриті та закриті. До перших належать ті, у яких існує невідповідність між запасами та потребами учасників: $\tilde{d} \neq 0$. Своєю чергою для других справедливий вираз: $\tilde{d} = 0$.

Якщо транспортна задача виявилася відкритого типу, тоді студент повинен виконати набір завдань 7–10 для її приведення до закритого. Спочатку треба визначитися з типом фіктивного учасника перевезень у рамках завдання 7:

— треба обрати варіант «постачальник», якщо запаси менше за потреби, тобто $\tilde{d} < 0$;

— значення «споживач» вважатиметься правильним в іншому випадку — коли запаси перевищують потреби, що записується як $\tilde{d} > 0$.

У завданні 8 треба вказати номер для створеного учасника транспортної задачі. Відповідь залежить від його типу — існує транзитивна залежність від значення $\tilde{d}$. На момент перевірки умови переходу до наступного завдання вважаємо, що оновлення кількостей учасників не відбувалося, тому можемо посилатися на значення m та n.

Завдання 9 присвячено визначенню запасів/потреб фіктивного постачальника/споживача. Відповідь дорівнює модулю різниці між загальними запасами та потребами, тобто $\left|\tilde{d}\right|$.

Під час виконання завдання 10 необхідно вказати, яке значення тарифів встановлюється для фіктивного учасника перевезень. Згідно з матеріалами дисципліни «Математичні методи дослідження операцій» це 0 [53, с. 152]. Після отримання правильної відповіді вважаємо, що постачальника/споживача успішно додано — відбувається оновлення пов'язаних змінних.

Завдання 11 присвячено обранню методу побудови опорного плану. Ініціалізуються значення змінних, де $\tilde{p}$ — індекс поточного плану перевезень, а $X^{(\tilde{p})}$ — множина кортежів координат комірок тарифів (номер постачальника, номер споживача), що належать базису. Варто зазначити, що дані в $\tilde{X}^{(\tilde{p})}$ — це метаінформація, необхідна для застосування методу потенціалів, а не опису самої транспортної задачі. Неправильних варіантів відповідей немає, за винятком порожнього вибору (коли $x(t)=\varnothing$). У разі переходу далі результат зберігається до пам'яті під символом $\tilde{M}$.

Після цього, в одному із завдань групи 12–14, студент повинен обрати базисну клітину. Якщо координати тарифу — номери постачальника та споживача — відповідають вимогам, що накладає метод $\tilde{M}$, тоді вони зберігаються всередині змінних $\tilde{s}$ і $\tilde{c}$ відповідно, після чого відбувається перехід далі.

У завданні 15 обчислюється необхідний обсяг поставки за обраною клітиною тарифу і зберігається в $\tilde{x}$. Користувач повинен ввести це число, і, залежно від того, чи дорівнюють запаси постачальника $A_{\tilde{s}}$ потребам споживача $B_{\tilde{c}}$, відбудеться перехід до однієї з гілок.

Якщо $a_{\tilde{s}} \neq b_{\tilde{c}}$, тоді студент виконуватиме завдання 16 та 17, де послідовно треба буде вказати оновлені значення запасів та потреб. В іншому випадку, коли $a_{\tilde{s}} = b_{\tilde{c}}$ — завдання 18, де потрібно ввести число 0.

$X^{(\tilde{p})}$ — це поточний план перевезень. Якщо теорема про ранг системи обмежень закритої транспортної задачі не виконується, тобто кількість знайдених базисних клітин менша за $m+n-1$, тоді повертаємося до групи завдань 12–14. Як тільки ця умова справджується — переходимо до завдання 22, де треба обчислити кількість базисних клітин, яку вимагає раніше згадана теорема. Вимоги до відповіді формально записані із використанням символів m та n для поліпшення сприйняття, оскільки це нагадує формулу обчислення рангу.

У завданні 23 треба ввести наявну кількість базисних клітин, після чого, у завданні 24, погодитися з тим, що теорема про ранг виконується. Дана ділянка діаграми потрібна лише для того, щоб нагадати студенту про важливість дотримання цієї теореми. Її виконуваність забезпечується такою логікою побудови завдань, яка унеможливлює втрату базисних комірок. До прикладу, якщо в разі постачання запаси постачальника відповідають потребам споживача, тоді відбувається розгалуження (завдання 20 та 21), залежно від того, якого учасника перевезень студент вирішив усунути з розгляду під час виконання завдання 19. У результаті проходження будь-якої з цих двох гілок отримаємо додаткову базисну комірку, визначену для того учасника, що залишився.

Якщо розв’язувана транспортна задача виявиться малою, тобто такою, де кількість постачальників або споживачів рівна 1, тоді застосування методу потенціалів не матиме сенсу. Причиною тому є відсутність вільних комірок, бо всі належатимуть опорному плану. У такому разі студент повинен виконати завдання 25, де має погодитися з тим, що рішення вже є оптимальним, після чого, у завданні 26, обчислити вартість перевезень.

В іншому випадку розпочинається застосування методу потенціалів. У завданні 27 треба обрати комірку потенціалу. Вибір користувача являє собою кортеж, що складається з двох елементів: номеру постачальника та споживача. Це робиться з лише з метою розрізнення, тому один із елементів завжди буде порожнім. На обрання клітини першого потенціалу обмеження не накладаються, а під час обрання наступних вимагається існування базисної клітини за учасником

перевезень протилежного типу. Результати успішного виконання зберігаються до змінних s_1 та c_1.

Завдання 28 присвячено введенню значення потенціалу. Якщо це виконується вперше, тоді коректною відповіддю є будь-яке ціле число. В іншому випадку правильна відповідь залежатиме від «парного» потенціалу, координати клітини якого (s_2, c_2) визначаються за процедурою, описаній на діаграмі. Завдання 27 та 28 разом утворюють цикл, вихід із якого відбувається, якщо ініціалізовані всі потенціали для поточної ітерації побудови плану перевезень $\tilde{p}$.

У рамках завдання 29 користувач має обрати клітини, що не належать базису. Їх координати являють собою кортежі з номерів постачальника та споживача, що не належать множині $\tilde{X}^{(\tilde{p})}$. Елементи правильної відповіді зберігаються до змінних $\tilde{s}$ та $\tilde{c}$.

Завдання 30 передбачає обчислення студентом характеристики попередньо обраної вільної клітини. Якщо ця операція виконана для всіх наявних вільних комірок — переходимо далі, інакше — повертаємося до завдання 29.

Під час виконання завдання 31 треба дати якісну оцінку отриманому плану перевезень за критеріями оптимальності та єдиності. Якщо рішення є оптимальним, тоді переходимо до завдання 26. щоб обчислити вартість транспортування вантажу. В іншому випадку відбувається перехід до завдань із застосування циклу переходу.

У завданні 32 користувач має побудувати замкнену ламану за кількома правилами. Перша вимога: відповідь є циклом. Це можна перевірити шляхом аналізу входжень номерів постачальників та споживачів із різними знаками. Нехай маємо вузли $(i_1, j_1), (i_2, j_2), \ldots, (i_l, j_l)$, де l — їхня кількість, $i_1, i_2, \ldots, i_l$ — номери постачальників, $j_1, j_2, \ldots, j_l$ — номери споживачів. Для кожного вузла ламаної призначаємо знаки: «+» — для непарних та «−» — для парних. Номер вузлів позначається k, причому це натуральні числа. Під час перевірки відповіді користувача формуємо 4 множини: P_1, P_2, N_1, N_2.

P_1 зберігатиме номери постачальників, які згадуються у вершинах, позначених знаком «+», а N_1 — зі знаком «−». Аналогічно для споживачів: P_2 та N_2. Для замкненого циклу справджуватиметься умова: $(|P_1 - N_1| = 0) \wedge (|P_2 - N_2| = 0)$.

Вираз $|\{i_k\} \cap \{i_{k+1}\}| + |\{j_k\} \cap \{j_{k+1}\}| = 1$ описує умову, що кожна пара послідовних вузлів ламаної має включати один спільний номер постачальника або споживача. У такий спосіб ланки кожного разу змінюватимуть напрямок на 90 градусів.

Останні вимоги описують, що побудову циклу переходу починаємо з вільної клітини, а інші вузли повинні розташовуватися за базисними. Варто підкреслити, що умова стосовно першого вузла є суворою: недостатньо, щоб деякий вузол зі знаком «+» був вільною клітиною.

Якщо побудована ламана правильна — вона зберігається до змінної $\tilde{C}$, а потім у рамках завдання 33 на її основі обчислюється обсяг перерозподілу $\tilde{\Delta}$, що повинен ввести студент.

Далі передбачається розгалуження залежно від кількості вузлів циклу зі знаком «−», за якими відповідні розміри поставок дорівнюють обсягу перерозподілу. Якщо лише один, тоді переходимо на початок застосування методу потенціалів — до завдання 27 зі скиданням змінних-лічильників та оптимізацією плану. В іншому випадку студент ще має обрати клітину, яка має покинути базис.

2.4 Математичні моделі оцінювання навчальних досягнень та прогресу

Діаграми станів (див. додатки А.1, А.2, А.3) вичерпно формалізують умови переходів між завданнями тренажера, утім їх недостатньо для опису процесу оцінювання навчальних досягнень та прогресу. Серед основних складнощів, пов’язаних із розробленням відповідної узагальненої моделі:

— неможливість попереднього обчислення кількості повторювань завдань: задача може мати декілька розв’язків або якщо дії користувача

впливають на кількість ітерацій (наприклад, для транспортних задач відомо, що опорний план, знайдений за методом мінімального тарифу, зазвичай кращій за той, що отриманий у результаті застосування методу північно-західного кута, а тому потребує меншої кількості кроків оптимізації);

— процес розв’язання згенерованої задачі далеко не завжди включає всі завдання, передбачені програмною реалізацією тренажера (деякі стають недосяжними через умови розгалужень), але це не повинно впливати на оцінку; приклад: згенерована транспортна задача закритого типу не потребує виконання завдань, присвячених введенню фіктивного учасника перевезень;
— ієрархічність підпроцесів розв’язання: цикли та розгалуження комбінуються між собою; за діаграмою станів тренажера з транспортних задач:
  — знаходження опорного плану є прикладом розгалуження всередині циклу;
  — оптимізація плану перевезень звичайної задачі, що не є малою — цикл у гілці розгалуження;
— різна вага завдань через неоднорідність їхньої важливості та складності; на прикладі транспортної задачі: вказання кількості постачальників вимагає менше часу, аніж обчислення їхніх загальних запасів;
— необхідність підтримки допоміжних елементів: вони не позначаються безпосередньо на діаграмі станів через рівень абстрагування, але потрібні для деталізованого опису змін у пам’яті під час переходів; приклад: додавання фіктивного учасника перевезень у транспортній задачі повинно супроводжуватися оновленням відповідних множин (запаси/потреби й тарифи), які з точки зору математики є незмінними.

Логіку формування послідовності завдань віртуальним тренажером називатимемо потоком. Його математичною моделлю є n-арне дерево F, кожен

вузол якого позначає окреме завдання або керувальну структуру: лінійну послідовність, розгалуження або цикл.

Лінійна послідовність — це обгортка для упорядкованого набору завдань або їхніх структур. Вона забезпечує послідовну передачу керування від одного елементу до іншого. За діаграмою (див. додаток А.1) до цього класу можна віднести набір завдань, присвячений приведенню транспортної задачі до закритого типу: спочатку треба вказати тип фіктивного учасника перевезень, після чого присвоїти йому порядковий номер, обчислити запаси/потреби та встановити тарифи. Є й інші приклади, зокрема додавання базисної клітини до опорного плану, визначення потенціалу для учасника перевезень та обчислення характеристики вільної клітини. До лінійних послідовностей також належить верхній рівень, до якого вкладаються всі завдання та структури, визначені в тренажері. Поодинокі завдання не обгортаються.

Розгалуження містить у собі щонайменше одну гілку з одним або кількома елементами. Під час роботи тренажера відбувається послідовний пошук тієї, для якої задовольняється пов’язана з нею умова. Якщо знайти не вдалося — структура вважається виконаною й керування переходить далі за правилами контейнера, у якому розташоване розгалуження. Приклади структур цього класу в тренажері, що проєктується: визначення необхідності приведення задачі до закритого типу, обрання методу знаходження опорного плану, встановлення потреби в оптимізації отриманого плану перевезень та інші.

Цикли забезпечують повернення керування до обгорнутого елементу, якщо вказана умова виконується. Прикладами є знаходження опорного плану перевезень, вказання потенціалів для учасників, обчислення значень характеристик вільних клітин та оптимізація розв’язку.

Нехай $w_{k,i}$ — ваговий коефіцієнт вузла дерева, де k — рівень вкладеності, i — локальний порядковий номер вузла (у межах рівня k). Для потоку завдань у цілому як найвищого рівня: $k = 0$ (корінь дерева), $i = 1$ (перший і останній елемент). Оскільки на цьому рівні лише один елемент, тоді вага $w_{0,1}$ — будь-яке

число, відмінне від 0. Надалі, на наступних рівнях, за допомогою вагових коефіцієнтів можна задавати співвідношення між значущістю різних складових потоку.

Оцінку $s_{k,i}$ у частках від одиниці за виконання елементу $f_{k,i}$ потоку можна обчислити за формулою:

$$s_{k,i} = \frac{\sum_{j=1}^{n} s_{k+1,j} \cdot w_{k+1,j}}{\sum_{j=1}^{n} w_{k+1,j}}, \qquad (2.13)$$

де n — кількість дочірніх вузлів.

Звідси маємо такі наслідки:

— цикл, для якого заздалегідь неможливо обчислити кількість ітерацій (типу while), як структура, вважається виконаною, коли умова переходу до наступної ітерації не задовольняється — отже, немає проміжних значень, які б забезпечували неперервне оновлення прогресу (на відміну від for);

— допоміжні елементи, що використовуються для деталізованого опису переходів між станами пам’яті, повинні мати вагу, що дорівнює 0 — отже, до батьківського контейнера висувається додаткова умова: наявність хоча б одного іншого елемента з ненульовою вагою.

На основі розробленої узагальненої моделі визначено кілька вузькоспеціалізованих: тренувальна та екзаменаційна — для задоволення потреб відповідних режимів, що постачатимуться разом із фреймворком для створення віртуальних тренажерів. Спеціалізація проводиться шляхом встановлення функціональної залежності оцінки виконання завдання від кількості неправильних відповідей m на елемент $f_{k,i}$, де кількість дочірніх $n = 0$.

Тренувальна модель толерантна до помилок користувача, а тому оцінка інваріантна до кількості помилок:

$$s_{k,i} = \begin{cases} 0, \text{ якщо не виконано,} \\ 1, \text{ якщо виконано.} \end{cases} \tag{2.14}$$

Екзаменаційна модель є більш суворою: наявність хоча б однієї помилки під час виконання окремого завдання зменшує максимальну оцінку за нього до 50% від початкової (без накопичення), тобто:

$$s_{k,i} = \begin{cases} 0, \text{ якщо не виконано,} \\ 0.5, \text{ якщо виконано } m \geq 1, \\ 1, \text{ якщо виконано та } m = 0. \end{cases} \tag{2.15}$$

Для візуалізації виокремлених структур завдань та запропонованого підходу до оцінювання розроблена спеціальна графічна нотація — n-арне дерево з керувальними вузлами. Результат її застосування (див. рис. 2.2). Для побудови використана програма draw.io.

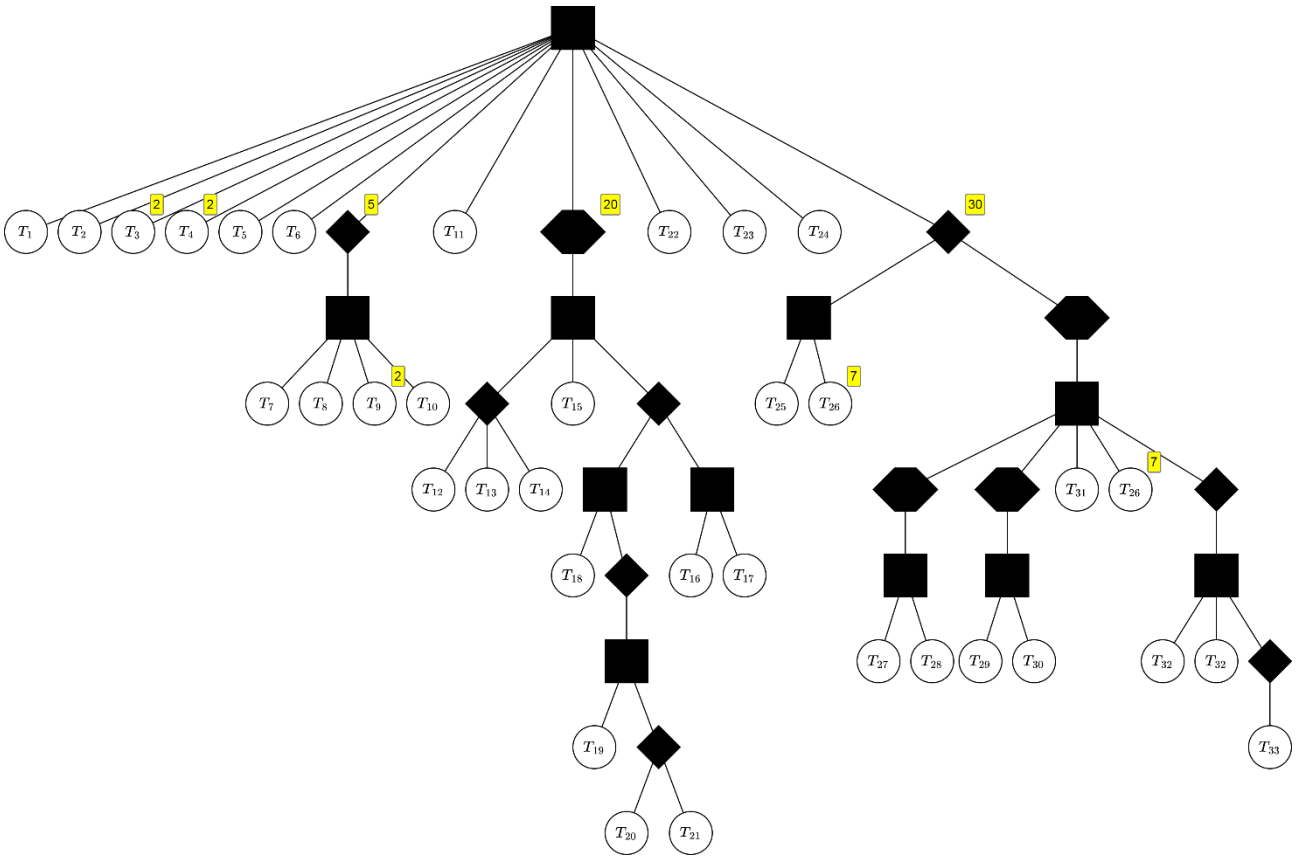


Рисунок 2.2 — n-арне дерево завдань з керувальними вузлами

Кожне завдання позначається кругом із написом T_i, де i — номер за діаграмою станів тренажера. За допомогою квадратів описуються лінійні послідовності. Ромби показують місця розгалужень. Цикли відображені у вигляді шестикутників.

Умовні позначення схожі на ті, що використовуються для створення блок-схем, не випадково. Асоціації дозволяють спростити інтерпретацію зображеного.

Для обходу дерева взятий за основу пошук у глибину із прямим порядком: спочатку відвідується корінь, після чого виконується послідовний рекурсивний обхід дочірніх вузлів. Керувальні вузли, що позначені чорним кольором, модифікують загальний алгоритм руху.

У лінійних послідовностях передача керування між дочірніми вузлами відбувається зліва направо одразу після завершення пошуку в глибину. Розгалуження передбачають відвідання щонайбільше одного дочірнього вузла. Цикли дозволяють почати обхід піддерева з початку.

Біля вузлів можна вказувати вагу. За замовчуванням вона дорівнює одиниці. У такий спосіб на діаграмі (див. додаток А.1) зазначено: обчислення загальної кількості запасів/потреб у 2 рази складніше за визначення кількості постачальників/споживачів, а трудовитрати на процес оптимізації (30) у півтора рази вищі за побудову початкового опорного плану (20). Такий підхід дозволяє більш точно оцінювати навчальні досягнення студентів у процесі роботи з тренажером.

2.5 Структурно-функціональне моделювання

IDEF0 (Integrated DEFinition for Function Modeling) — методологія моделювання функцій. Її графічна нотація складається з блоків, що позначають процеси, та стрілок, що показують:

— входи — інформація або ресурси, необхідні для виконання функції;

— виходи — результати роботи функції;

— механізми — інструменти виконання функції;

— керування — засоби, які впливають на виконання функції.

За допомогою декомпозиції можна аналізувати систему в розрізі різних рівнів деталізації, тим самим спрощуючи ідентифікацію основних процесів та залежностей між ними. Також діаграми в нотації IDEF0 демонструють, у який спосіб відбувається обробка інформації шляхом визначення потоків її руху як всередині системи, так і обміну із зовнішнім середовищем. Дана методологія — це ефективний спосіб документування процесів, оскільки він стандартизований.

Структурно-функціональне моделювання системи — віртуального тренажера, розробленого на основі запропонованої інформаційної технології — проводилося в середовищі програми AllFusion Process Modeler r7. Основним процесом є «Робота віртуального тренажера» (див. рис. 2.3). На вхід подається режим роботи. У якості механізмів — студент, адміністратор, апаратне та програмне забезпечення. Керування процесом відбувається на основі законодавства, нормативних документів закладу вищої освіти та силабусу дисципліни. Згенерована задача, оцінка навчальних досягнень є виходами системи.

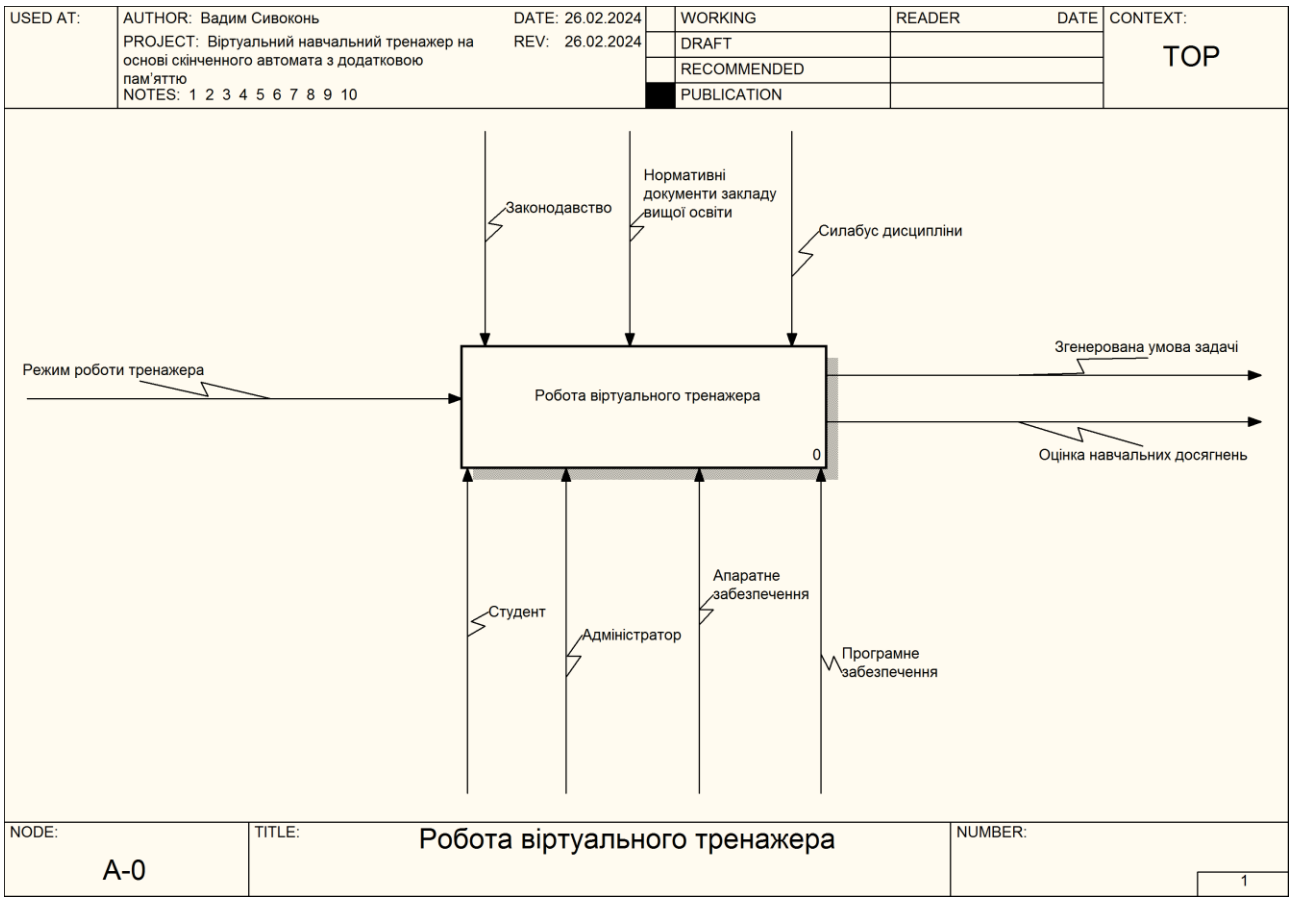


Рисунок 2.3 — Контекстна діаграма процесу «Робота віртуального тренажера» в нотації IDEF0

У результаті декомпозиції процесу «Робота віртуального тренажера» виокремлено 4 підпроцеси (див. рис. 2.4):

— обрання режиму роботи: визначається із якими налаштуваннями в подальшому працюватиме тренажер. На вхід подається бажаний режим роботи. Механізмами є студент, адміністратор, апаратне та програмне забезпечення. Процес керується законодавством (наприклад, Законом України «Про захист персональних даних»). Виходом є налаштування генерації задачі;

— генерація умови задачі: випадковим чином формується умова задачі. На вхід приймає попередньо сформовані налаштування. Механізми: адміністратор, апаратне та програмне забезпечення. Нормативні документи закладу вищої освіти та силабус дисципліни контролюють процес генерації. На виході — згенерована умова задачі, яку можна вважати як

входом до наступного процесу, так і окремим результатом роботи системи;

— виконання завдань: передбачає покрокове розв’язання задачі. Входом є раніше згенерована умова задачі. Серед механізмів: студент, апаратне та програмне забезпечення. Процес контролюється нормативними документами закладу вищої освіти та силабусом навчальної дисципліни. Виходом є інформація про відповіді студента;

— завершення роботи: підбиває підсумки навчальної діяльності. На вхід надходить інформація про відповіді студента на завдання тренажера. Механізмами є адміністратор, апаратне та програмне забезпечення. Процес керується законодавством. На виході — оцінка навчальних досягнень.

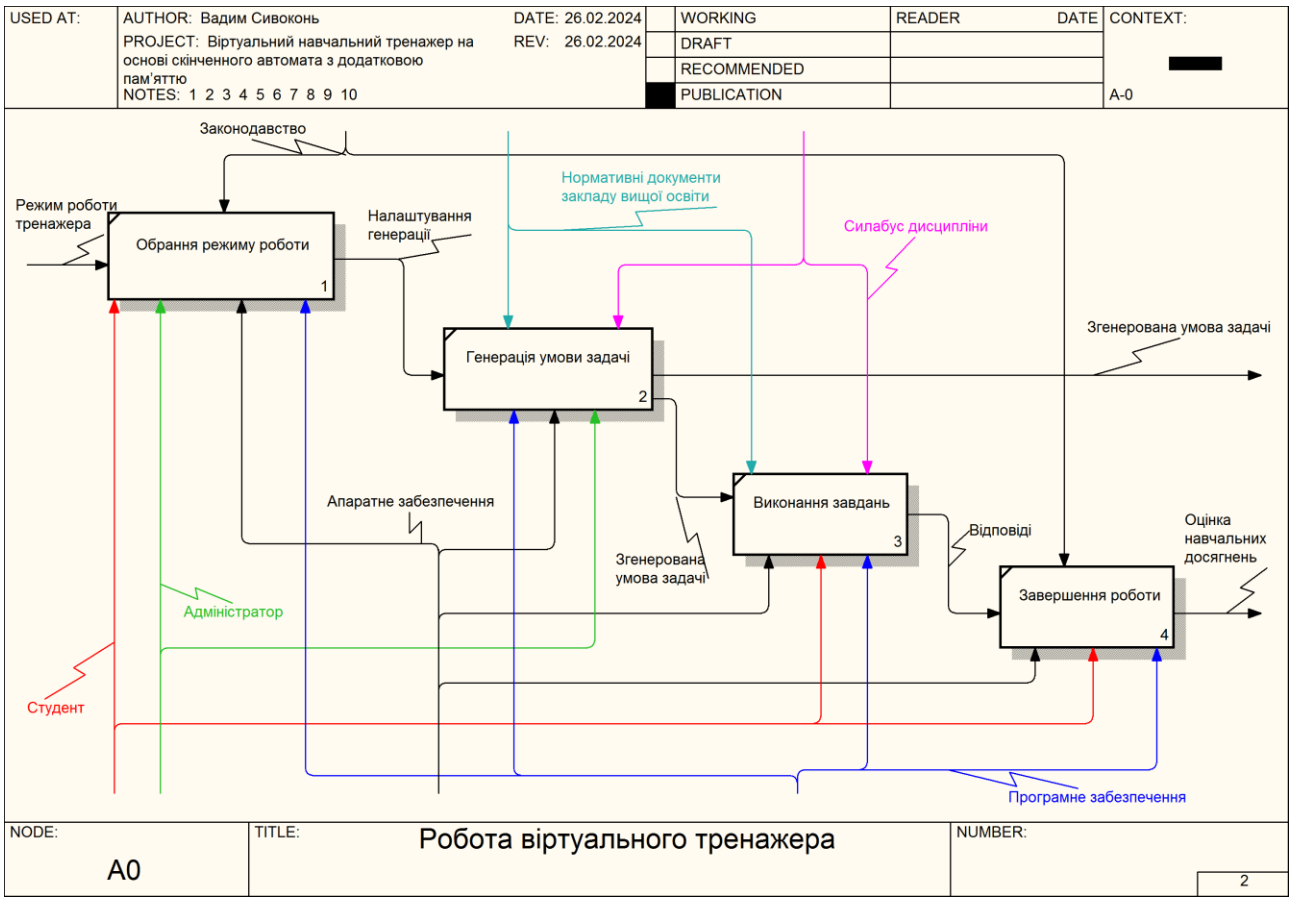


Рисунок 2.4 — Діаграма декомпозиції процесу «Робота віртуального тренажера» в нотації IDEF0

Декомпозиція першого рівня забезпечила достатній для наших потреб рівень деталізації функціонально-структурної моделі віртуального тренажера.

2.6 Моделювання варіантів використання

UML — це уніфікована мова моделювання, що використовується для проєктування систем на основі стандартизованих засобів візуалізації. Вона допомагає описувати структуру та поведінку застосунків на етапі створення документації у зрозумілій формі. Завдяки абстрагуванню від технічних подробиць дозволяє отримати загальну картину проєкту.

Діаграма варіантів використання (або діаграма прецедентів) — поведінкова діаграма в графічній нотації UML, за допомогою якої на високому рівні можна описати способи взаємодії з системою. Крім опису функціональних вимог до застосунку відбувається ідентифікація зовнішнього впливу. Спрощує написання сценаріїв тестування. Також на її основі створюється документація для користувачів.

Основними елементами діаграм цього типу є актори, варіанти використання, відносини та система. Актори — це сутності поза системою, що взаємодіють із нею за вказаними у вигляді варіантів використання сценаріями для отримання значущого результату.

За допомогою відносин задаються зв’язки між елементами діаграми. Актори з варіантами використання пов’язуються асоціаціями. Між варіантами використання також встановлюються зв’язки, основними серед яких залежність включення та залежність розширення. Перші дозволяють описати окремі кроки варіанту використання, що їх включає. Другі призначені для опису додаткової логіки, яка виконуватиметься за потреби, у такий спосіб розширюючи базовий варіант. Інколи для спрощення діаграми застосовуються узагальнення.

Для системи, тренажера з транспортних задач, побудована діаграма варіантів використання (див. рис. 2.5) із застосуванням draw.io, що має потужну підтримку цієї графічної нотації.

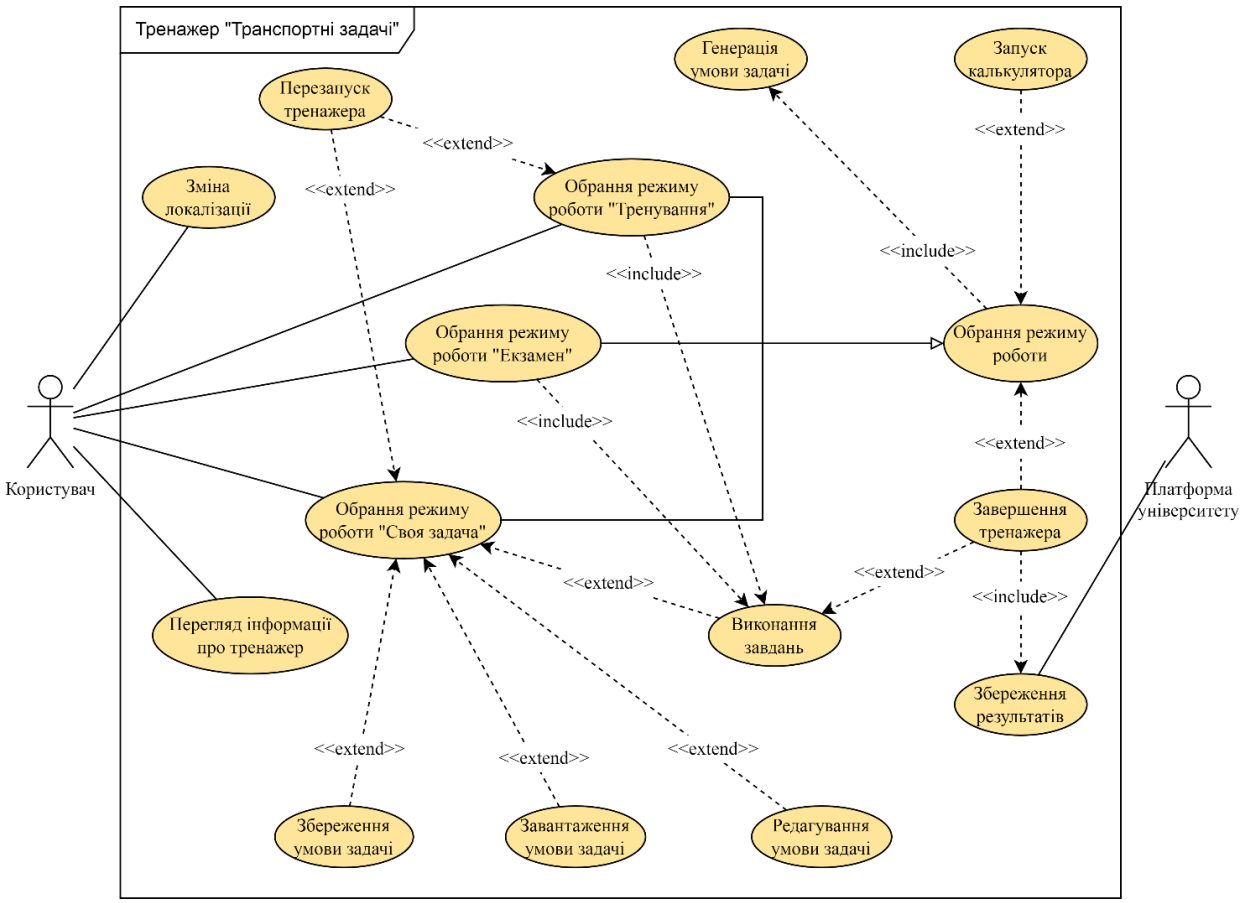


Рисунок 2.5 — Діаграма варіантів використання тренажера

Акторами є користувач та платформа університету. Під першим насамперед маємо на увазі студентів, які працюватимуть із цим тренажером. Другий позначає сайт університету, до якого відбуватиметься інтеграція: це можуть бути Mix або Lectur.ED. Адміністратор, як актор, не зазначається, оскільки викладачі не мають механізмів віддаленого налаштування тренажерів на сервері. Можна тільки завантажувати передналаштовані застосунки та вказувати параметри навчального об’єкта, які пропонує платформа, наприклад, максимальна оцінка завдання в балах.

За варіантами використання видно, що користувач може працювати в різних режиму тренажера: «Тренування», «Своя задача» та «Екзамен». Їх спільна логіка описана за допомогою батьківського варіанту використання «Обрання режиму роботи». Генерація задачі відбувається автоматично незалежно від режиму. Можна користуватися вбудованим калькулятором. Тренажер

дозволяється завершити в будь-який момент — результати надсилатимуться на сервер університету.

Крім того є певні особливості окремих режимів. Перезапуск тренажера доступний в усіх, крім екзамену. У режимі «Своя задача» користувач може зберегти умову або завантажити власну. До того ж є опція редагування прямо в додатку. Якщо проводити декомпозицію, то рівнем нижче знаходяться можливості оновлення запасів/потреб постачальників/споживачів, додавання та видалення учасників перевезень, зміна тарифів, а також повна перегенерація умови транспортної задачі.

Одразу до виконання завдань користувач переходить у режимах «Тренування» та «Екзамен», а у «Своя задача» — коли вважатиме за потрібне. Після виконання всіх завдань, тренажер завершиться автоматично.

Користувач може змінювати локалізацію тренажера та переглядати інформацію про нього.

2.7 Засоби реалізації

Програмною реалізацією нової інформаційної технології є фронтенд-фреймворк для створення віртуальних тренажерів у вигляді односторінкових вебдодатків. Його поточна версія являє собою шаблон із базовими класами та ресурсами, необхідними для розроблення інтерактивних практичних завдань під екосистему навчальних сервісів Mix та Lectur.ED Сумського державного університету.

Особливістю розроблення віртуальних тренажерів, зокрема в нашому університеті, є дотримання принципу Desktop First. Насамперед навчальний процес здебільшого спирається на роботу саме з комп’ютером або ноутбуком. Мобільні пристрої володіють потужним функціоналом, утім для деяких завдань є незручними. Основна причина — порівняно невеликий розмір екрану, що може негативно впливати на зручність та продуктивність використання, особливо коли

застосунок має складний інтерфейс. Крім того пристрої володіють нижчою швидкодією.

JavaScript — це високорівнева інтерпретована мультипарадигмальна мова програмування загального призначення. Вона є реалізацією специфікації ECMAScript. JavaScript часто використовується для створення як клієнтської, так і серверної частини додатків. У браузері ця мова є інструментом інтерактивізації вебсторінок: дозволяє керувати DOM-елементами, проводити валідацію даних, обробляти події та організовувати обмін інформацією із сервером. На боці сервера може опрацьовувати HTTP-запити клієнтів, звертатися до бази даних, взаємодіяти з іншими API, генерувати контент, керувати сесіями користувачів та іншими процесами.

Для запуску сценаріїв необхідне середовище виконання. Ним може виступати браузер, де рушії JavaScript займаються інтерпретацією. У подальшому, зі зростанням обсягів коду для виконання, перейшли до JIT-компіляції. Серед найбільш відомих рушіїв JavaScript: V8 у Google Chrome, SpiderMonkey у Firefox та ChakraCore в Microsoft Edge.

Їхнє застосування не обмежується лише браузерами. До прикладу, на рушії V8 побудовані такі серверні середовища виконання JavaScript, як Node.JS та Deno. Також він використовується в основі фреймворку для створення настільних додатків Electron.

Небраузерні середовища виконання потрібні для фронтенд-розробки. Вони дозволяють керувати залежностями проєкту та використовувати додаткові інструменти, такі як збиральник коду, локальний вебсервер, лінтер тощо.

Найпопулярнішими небраузерними середовищами виконання JavaScript є Node.js, Deno та Bun. Перевагами першого є перевіреність часом та багата екосистема пакетів, що містить велику кількість бібліотек та фреймворків. Друге середовище пропонує вбудовану підтримку TypeScript, покращену модель безпеки із пісочницею та правами доступу, а також керування залежностями на основі URL-адрес. Третє має кращу продуктивність у порівнянні з іншими: виконання

коду та робота із залежностями відбувається швидше. До того ж Bun є збиральником коду, що позбавляє від необхідності встановлювати додаткові інструменти, такі як Webpack [56].

За останніми доступними результатами опитування для розробників «State of JavaScript», що проводилося в 2022 році (див. рис. 2.6), першість здобув Node.js зі значним відривом: він отримав 27974 голоси, у той час як Deno набрав 3344, а Bun — 1263. Це свідчить про те, що для респондентів переваги Node.js у вигляді тривалої історії успішного розвитку та розвиненої екосистеми пакетів є важливішим. З тієї ж причини це середовище виконання JavaScript обране для розроблення програмної реалізації технології.

Рисунок 2.6 — Використання середовищ виконання JavaScript [57]

Для підтримки додаткових можливостей розроблення з’являлися мови, що компілюються в JavaScript. Наприклад, CoffeeScript пропонує коротший синтаксис, а TypeScript запроваджує статичну типізацію. У рамках опитування «State of JavaScript» 2022 року (див. рис. 2.7), лідерство здобув TypeScript: за нього

проголосували 26986 учасників. Найближча альтернатива, Elm, отримала лише 624 голоси.

Рисунок 2.7 — Популярність мов, що компілюються в JavaScript [57]

TypeScript — вдалий вибір для великих масштабованих проєктів, зокрема фреймворків. За рахунок статичної типізації код стає більш передбачуваним та зрозумілим для розробників, що значно спрощує його супровід. Розширена підтримка з боку IDE у вигляді статичного аналізатора, автодоповнення та інструментів рефакторингу дозволяють пришвидшити процес розроблення. Доповнення синтаксису надають можливість повноцінно оперувати концепціями ООП, такими як абстрактні класи, інтерфейси та узагальнені типи. Завдяки активній спільноті існує велика кількість бібліотек та фреймворків, готових до використання. Враховуючи наведені переваги, TypeScript був обраний у якості мови написання програмної реалізації в рамках кваліфікаційної роботи.

Для керування залежностями потрібний менеджер пакетів. Разом з обраним середовищем Node.js можна використовувати npm, yarn або pnpm.

Перший — це пакетний менеджер за замовчуванням, що постачається з Node.js; є найбільш сумісним із цим середовищем та, відповідно, популярним. Він встановлює залежності ієрархічно: папка «node_modules» створюється для кожного пакету. Так за рахунок використання додаткового дискового простору відбувається ізоляція залежностей та скорочення кількості можливих конфліктів версій.

Yarn — альтернатива від компанії Facebook. Він підтримує паралельне завантаження пакетів, що скорочує часові витрати на завантаження та оновлення залежностей. Застосування пласкої моделі встановлення пакетів дозволяє вирішити проблему дублювання. У такий спосіб можна дещо скороти використання дискового простору, утім із ризиком появи конфліктів. Також yarn надає змогу встановлювати пакети без підключення до мережі Інтернет.

Третій, pnpm, позиціонується як удосконалена версія npm. Для економії дискового простору використовує загальне сховище пакетів та жорсткі посилання. Модель встановлення залежностей є ієрархічною, тому дублювання можливі, як і в npm [58].

За допомогою сервісу sotagtrends.com побудовані графіки кількості використань тегів із назвами пакетних менеджерів у питаннях на Stack Overflow, платформі запитань-відповідей для розробників, за місяцями (див. рис. 2.8). Варто зазначити, що під yarn та yarnpkg мається на увазі один і той самий пакетний менеджер. У такий спосіб вдалося опосередковано оцінити популярність цих інструментів.

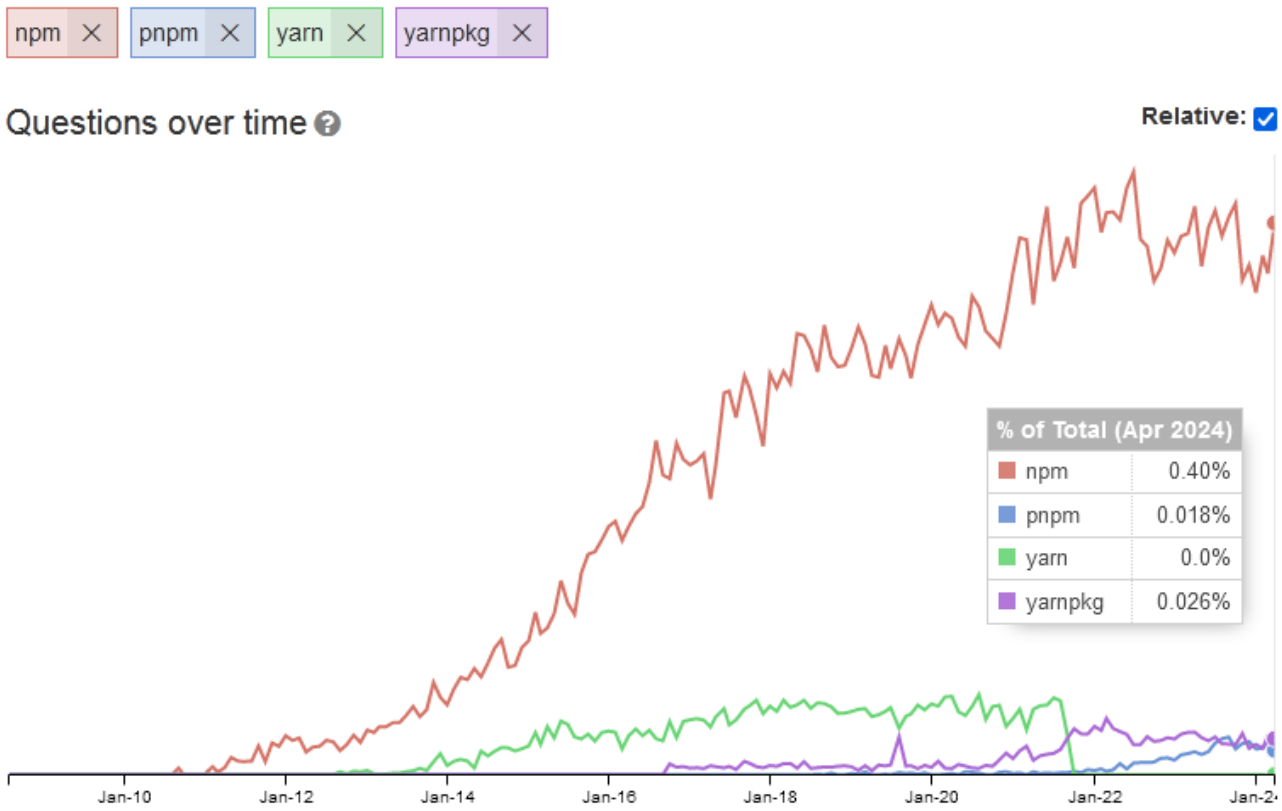


Рисунок 2.8 — Кількість запитань щодо пакетних менеджерів на платформі Stack Overflow [59]

Від початку свого існування npm тримає лідерство. Ймовірно, це пов’язано з тим, що він є менеджером пакетів для Node.js за замовчуванням, а тому не потребує додаткових дій зі встановлення. Прийнято рішення про те, що для розроблення програмної реалізації використовуватиметься npm, оскільки можливостей цього інструменту достатньо для виконання поставлених завдань.

У фронтенд-розробці використовуються бібліотеки та фреймворки, коли виникає необхідність створення складного динамічного користувацького інтерфейсу та ефективного керування станом вебзастосунку. Найперше такий підхід дозволяє зекономити час за рахунок застосування перевірених оптимізованих рішень типових завдань. До того ж поліпшується підтримка проєкту завдяки стандартизації використовуваних підходів. На додаток бібліотеки та фреймворки зазвичай мають деталізовану документацію, що поліпшує процес розроблення. Ба більше, часто з’являються плагіни, які розширюють можливості цих технологій.

Найбільш відомими фронтенд-фреймворками на час написання кваліфікаційної роботи були Angular та Vue.js, а бібліотекою — React.

В Angular завдяки підтримці двостороннього зв’язування зміни в інтерфейсі автоматично відображаються в моделі та навпаки. Механізм ін’єкції залежностей знижує зв’язність модулів — полегшується їх тестування та перевикористання. Основною мовою за замовчуванням є TypeScript. Суттєвим недоліком є крута крива навчання. Високий поріг входження створюється необхідністю розуміння ключових концепцій, на основі яких працює Angular, серед яких ін’єкція залежностей, потоки даних, компоненти, модулі, сервіси, директиви та роутинг. Ще одним недоліком у порівнянні з іншими технологіями є написання великої кількості шаблонного коду. До того ж розмір файлів фреймворку є порівняно великим, що негативно впливає на час завантаження вебзастосунку в браузері.

Vue.js є високопродуктивним фреймворком завдяки малому розміру та використанню віртуальної моделі документа. Модель передачі даних є двоспрямованою: від батьківського компонента до дочірнього й навпаки. За рахунок масштабованості та гнучкості він підходить для різних за величиною проєктів. Фреймворк пропонує простий синтаксис у вигляді комбінації можливостей шаблонів HTML (тег <template>), стилів CSS (тег <style>) та скриптів JavaScript (тег <script>). У такий спосіб розроблення компонентів є інтуїтивно зрозумілим. Серед недоліків фреймворку виділяють надмірну гнучкість, що може стати причиною неконсистентного коду. До того ж екосистема розширень недостатньо розвинута, ймовірно, через мовний бар’єр: основна спільнота розробників знаходиться в Китаї.

React — це бібліотека JavaScript від компанії Facebook, що дозволяє створювати комплексні динамічні інтерфейси. В основі використовується віртуальна модель документа. Для опису зовнішнього вигляду та логіки функціонування компонентів пропонується синтаксис JSX/TSX, що є сумішшю JavaScript/TypeScript та XML. Він слугує лаконічною заміною конструкції «React.createElement». Бібліотека працює за односпрямованою моделлю передачі

даних: від батьківського компонента до дочірнього. Розробники React забезпечують зворотну сумісність із попередніми версіями, завдяки чому оновлення залежностей проєкту не спричиняє проблем. Активна спільнота створила на базі цієї бібліотеки велику кількість розширень, що пришвидшують процес розроблення. Серед недоліків технології: необхідність вивчення особливостей синтаксису опису компонентів JSX/TSX. Відсутність сталих конвенцій щодо правильної реалізації типових аспектів застосунку може ускладнювати розуміння логіки його роботи.

За статистикою запитань на Stack Overflow щодо використання цих технологій першість здобув React (див. рис. 2.9). Схожі результати отримані в 2022 році під час аналізу вимог до понад 60000 вакансій, розміщених на сайті з працевлаштування indeed.com: у 78.1% випадків вимагалися знання React, у 21.1% — Angular та в 0.8% — Vue.js [60].

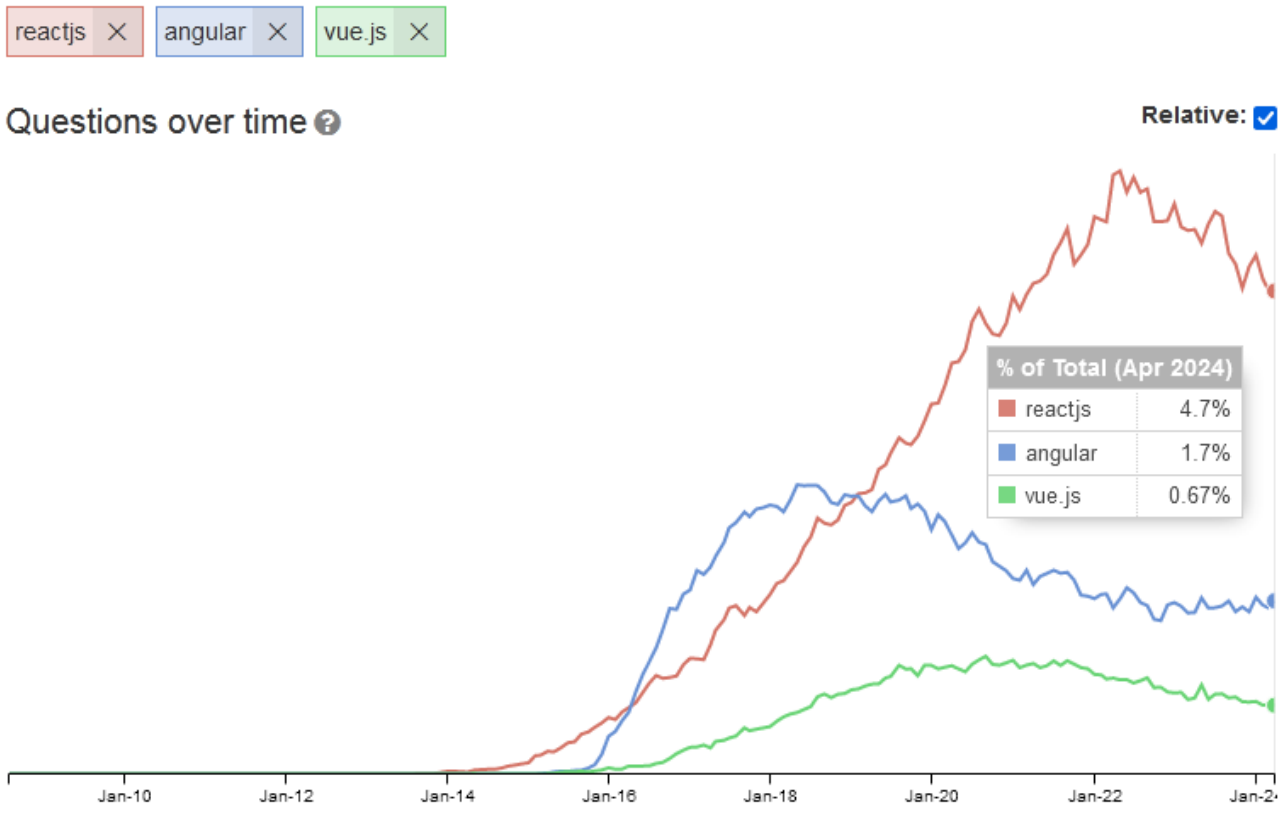


Рисунок 2.9 — Кількість запитань щодо фронтенд-бібліотек та фреймворків на платформі Stack Overflow [61]

Вибір зроблений на користь React насамперед з огляду на його потужну підтримку в колі веброзробників. За її використання маємо більше шансів знайти потрібні плагіни. До того ж односпрямована модель передачі даних є простішою в порівнянні з двоспрямованими, завдяки чому логіка функціонування більш передбачувана. Це дозволяє скоротити час налагодження. React надає великий простір свободи для побудови внутрішньої архітектури застосунку, що особливо важливо для розроблення фреймворку зі створення віртуальних тренажерів: можна вільно вводити підтримку необхідних концепцій.

Для створення проєкту на основі React потрібно виконати ряд підготовчих дій. Усе починається зі створення файлу опису проєкту package.json шляхом ініціалізації пакету.

Наступний етап — встановлення залежностей. Вони бувають кількох типів. Найбільш поширеними серед них є залежності часу виконання, без яких додаток не може функціонувати, та залежності, необхідні винятково для його розроблення. У package.json перші позначаються як «dependencies», а другі — як «devDependencies».

До переліку основних залежностей належать пакети react та react-dom. Перший є ядром бібліотеки: у ньому зосереджений функціонал для підтримки життєвого циклу компонентів, керування станами, передачі властивостей, застосування хуків, використання віртуальної моделі документа тощо. Другий необхідний для рендерингу компонентів у браузері. Причиною поділу бібліотеки на react та react-dom є принцип розділення відповідальності, що дозволяє під час розроблення під мобільні пристрої на основі React Native замість react-dom використати react-native. У такий спосіб досягається перевикористання пакету react-dom із ключовим функціоналом.

Своєю чергою «devDependencies» поліпшують процес розроблення. Серед найбільш затребуваних виділяють транскомпілятори JavaScript, інструменти створення збірок (пакувальники), локальні вебсервери, компілятори мов-розширень, статичні аналізатори коду, плагіни та визначення типів.

Транскомпілятори JavaScript дозволяють перетворювати код, написаний із використанням «синтаксичного цукру», на сумісний із ширшим спектром версій рушіїв. Їх функціонування не обмежується виключно специфікацією ECMAScript: це також стосується JSX/TSX, що передбачає перетворення XML-подібної розмітки на вкладені один до одного виклики React.createElement.

Серед найбільш відомих транскомпіляторів виділяють TypeScript, Babel, SWC та esbuild. Перший належить до цього переліку за сумісництвом, оскільки у файлі налаштувань tsconfig.json можна вказати цільову (target) версію JavaScript, до якої буде приводитися перетворений код. Зазвичай не використовується за межами проєктів на TypeScript.

Навколо Babel силами активної спільноти розробників вибудована потужна екосистема плагінів, що шляхом модифікації абстрактного синтаксичного дерева розширюють можливості інструмента: вводиться підтримка додаткових синтаксичних конструкцій (наприклад, JSX), проводиться оптимізація вихідного коду [62].

Спільним недоліком TypeScript та Babel є проблеми зі швидкодією транспіляції. Наявні альтернативи, SWC та esbuild, виконують її в декілька разів швидше. Різниця обумовлена використанням мов програмування нижчого рівня (Rust та Go відповідно) та паралельністю виконання на кількох ядрах процесора. Це підтверджують результати бенчмаркінгу [63]. Першість здобув esbuild. За ним слідує SWC із мінімальним відривом. Також встановлено, що розмір проєкту може впливати на ефективність інструменту транспіляції, що наочно прослідковується на прикладі TypeScript, що працює повільніше за Babel на простих проєктах (див. рис. 2.10).

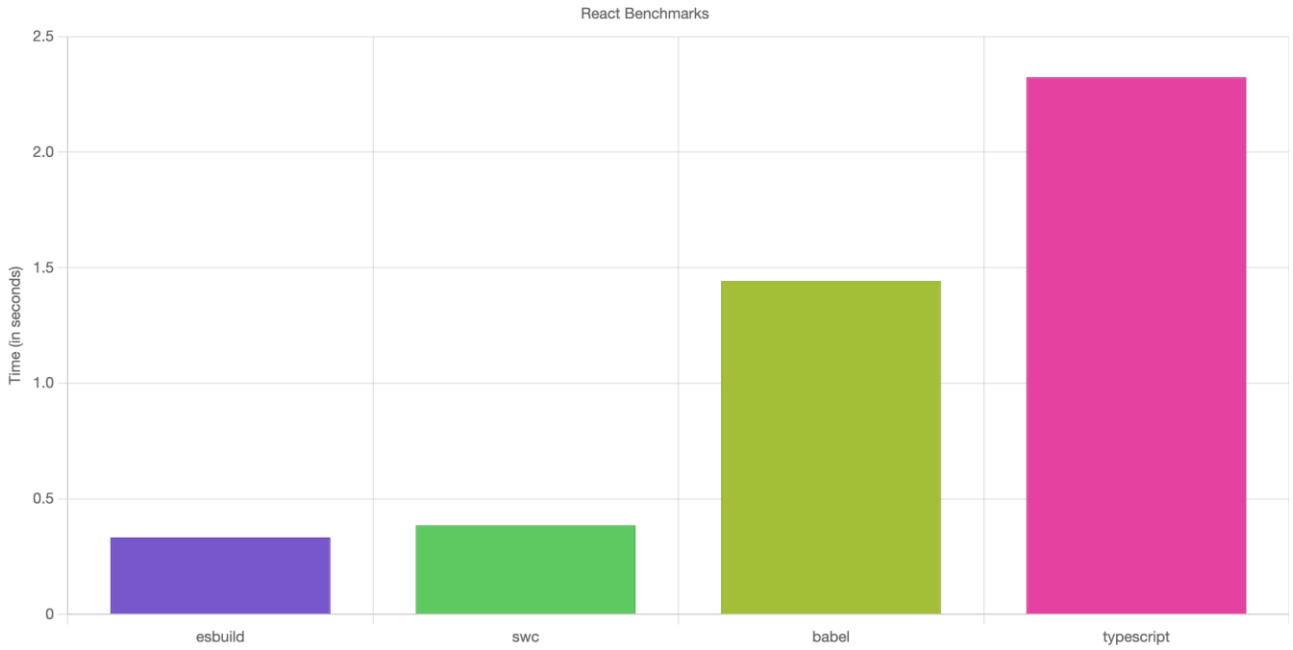


Рисунок 2.10 — Час транспіляції малих проєктів різними транскомпіляторами [63]

Недоліком SWC та esbuild є їх відносна новизна на ринку: можуть бути відсутні потрібні аналоги плагінів, що наявні для Babel.

За допомогою пакувальників із модулів утворюються збірки. Пріоритетом функціонування цих інструментів є ефективне керування ресурсами. Для зниження кількості запитів на сервер сценарії JavaScript та описи стилів CSS об’єднуються в збірки, проводиться вбудовування невеликих зображень. Методи оптимізації, як-от tree shaking та мініфікація, дозволяють зменшити розміри ресурсів шляхом видалення зайвого без порушення правильної інтерпретованості.

Найбільш поширеними пакувальниками є webpack, esbuild, rollup та Parcel.

Webpack надає можливість налаштувати гнучкі правила збирання проєкту. Для підтримки різних типів файлів використовуються відповідні завантажувачі. За допомогою плагінів можна налаштовувати додаткові дії, що повинні виконатися під час збирання, наприклад, мініфікація коду, генерація HTML-документів за шаблонами тощо. Недоліками є складність конфігурування та повільне збирання великих проєктів [64].

Esbuild — популярна альтернатива Webpack із надзвичайною швидкодією. За результатами бенчмаркінгу (див. рис. 2.11) за допомогою інструменту проведено побудову production-збірки великого проєкту, що складався з 10 копій бібліотеки three.js, усього за 0.39 с, причому зі звичайними налаштуваннями: виконання мініфікації коду та генерація карт ресурсів. Це в декілька разів краще за роботу parcel 2 (14.91 с), rollup 4 у зв'язці з terser для мініфікації (34.10 с) та webpack 5 (41.21 с).

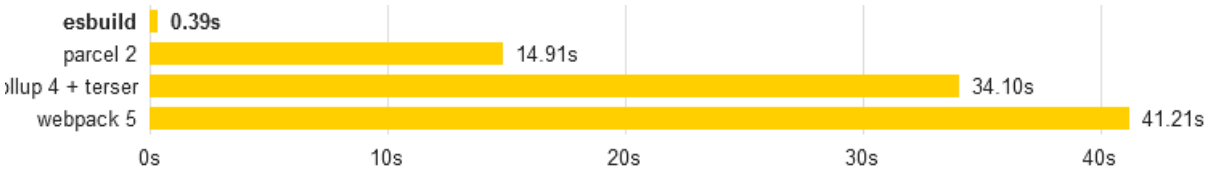


Рисунок 2.11 — Час створення production-збірки [65]

Інструмент має вбудовану підтримку транскомпіляції звичайного JavaScript, розширення TypeScript, синтаксису JSX для опису React-компонентів та CSS. Можна гнучко налаштовувати не тільки цільовий стандарт, а й версії браузерів. Це перспективна технологія, утім знаходиться на початку свого розвитку, про що свідчить версія продукту (0.20.2 станом на 05.04.2024). Інструмент постійно оновлюються: опції можуть як додаватися, так і видалятися. З огляду на це рекомендується триматися обраної версії продукту [66]. Наявна підтримка плагінів, проте екосистема ще не достатньо розвинута.

Rollup — збиральник, що спирається на використання можливостей модулів ECMAScript. Інструмент зазвичай створює збірку дещо меншу за розміром та накладними витратами, якщо порівнювати з результатами конкурентів. Rollup має гнучку систему плагінів, за допомогою яких розширюються можливості збиральника: оптимізація ресурсів, підтримка синтаксису стилів препроцесорів тощо. Недоліком є складність підтримки старих версій браузерів: це потребує додаткових налаштувань. До того ж спільнота не настільки велика, як у webpack, що позначається на виборі плагінів [35].

Parcel — новий збиральник, ідеєю якого є нульова конфігурація, тобто вбудована підтримка необхідних технологій без потреб у плагінах і додаткових налаштуваннях. Недоліком Parcel є обмежений контроль процесу збирання: неможливо легко змінити поведінку за замовчуванням. Є проблеми з підтримкою деяких фреймворків та бібліотек [67].

Локальні вебсервери дозволяють організувати процес розроблення в реальному часі: одразу можна бачити, як зміни в коді впливають на інтерфейс та поведінку застосунку. Це зручні інструменти для налагодження та тестування. Зазвичай вони постачаються разом з інструментами для збирання проєкту.

Мови-розширення створюються не тільки для JavaScript — додаткових можливостей потребують й інші технології, зокрема CSS. Для неї створюються інструменти двох видів: препроцесори та постпроцесори. Перші підтримують компіляцію «синтаксичного цукру» у звичайні описи стилів. Другі пропонують застосування різних правил обробки під час парсингу вже готових файлів CSS: мініфікація, додання вендорних префіксів тощо.

До найбільш популярних препроцесорів можна віднести SASS, LESS та Stylus. В останні роки беззаперечне лідерство демонструє SASS, про що свідчать результати опитування «State of CSS 2023» щодо використовуваних препроцесорів та постпроцесорів (див. рис. 2.12).

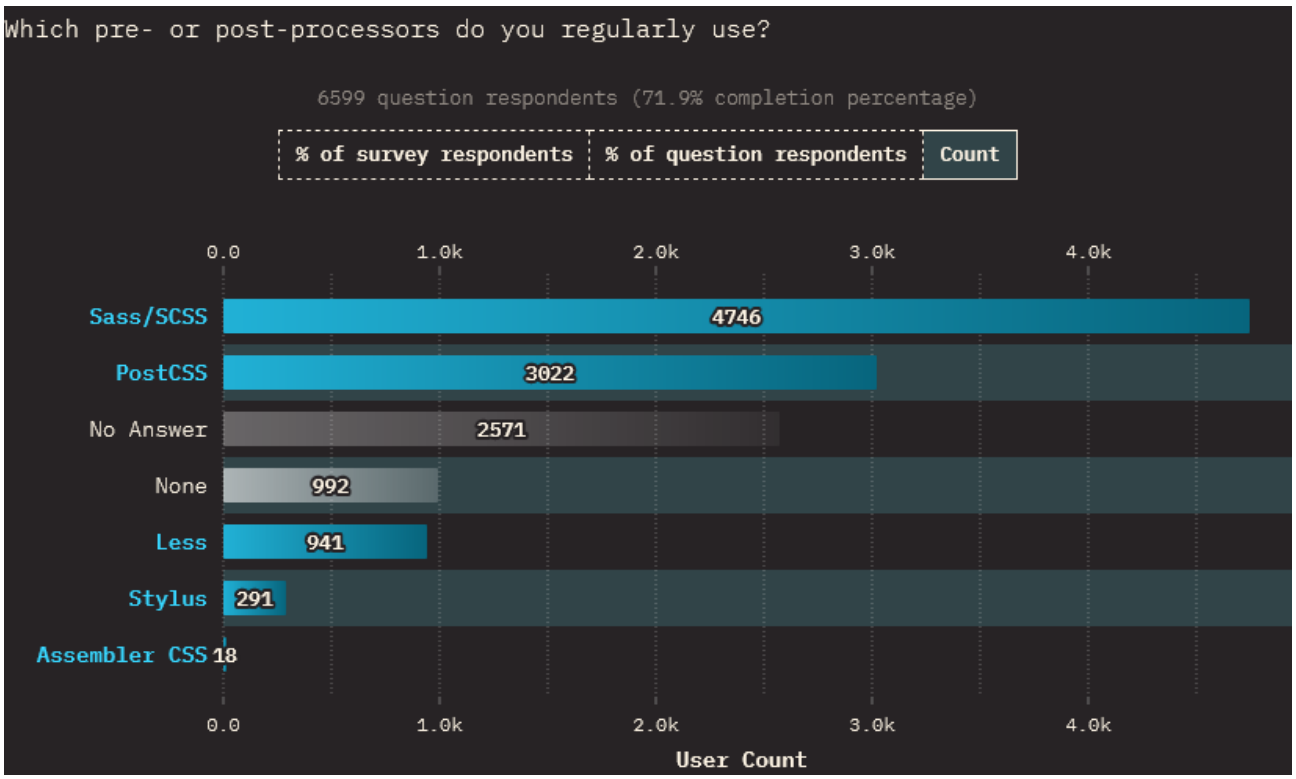


Рисунок 2.12 — Використання пре- та пост-процесорів для CSS [68]

SASS пропонує два синтаксиси: із використанням фігурних дужок і крапок з комою (файли стилів мають розширення «.scss») або без них («.sass»). Дозволяється визначати змінні, значення яких обчислюватимуться в момент компіляції та підставлятимуться в місця використання. Типи змінних не обмежується числами — це можуть бути рядки, коди кольорів, списки, логічні значення, null, асоціативний масив або посилання на функцію [69]. Для створення повторюваних фрагментів коду використовуються цикли. Підтримується умовна компіляція завдяки використанню розгалужень. До того ж можна визначати вкладені стилі. Міксини відіграють роль шаблонів, що можуть бути параметризованими. Для уникнення дублювання вихідного коду підтримується спадкування селекторів. Також запроваджується модульність, що поліпшує навігацію між стилями [70].

Інші препроцесори, LESS та Stylus, володіють аналогічним функціоналом, утім незначною мірою відрізняється синтаксис.

Статичні аналізатори зазвичай використовуються для виявлення помилок на ранніх етапах та підтримання встановленого стилю написання коду. Деякі здатні знаходити потенційно небезпечні ділянки на предмет можливості SQL-ін’єкцій, XSS-атак та інших загроз безпеці. Також можуть висувати пропозиції щодо рефакторингу для впровадження кращих практик, що набуло актуальності на тлі розвитку технологій штучного інтелекту.

Під час створення фронтенд-застосунку розробники переважно користуються ESLint та Prettier. Перший дозволяє знаходити потенційні проблеми в коді. Інколи він навіть може запропонувати їх автоматичне виправлення. Другий слідкує за дотриманням вимог до оформлення коду. Він доповнює перелік правил, які перевіряє ESLint.

Ці два інструменти інтегруються з різними середовищами розробки. У такий спосіб наочні підказки для розробника формуватимуться в режимі реального часу.

Обидві технології мають якісну документацію, що полегшує процес налаштування. Для ESLint та Prettier створено багато плагінів, що підвищують корисність статичного аналізу.

Інші «devDependencies» слугують доповненням до раніше зазначених. Пакети із визначеннями типів потрібні для функціонування механізму автодоповнення в IDE для проєктів на TypeScript. Плагіни використовуються для розширення можливостей інструментів, наприклад, пакувальників, щоб мати змогу працювати зі специфічними типами файлів.

Щоб усі розглянуті інструменти злагоджено працювали, необхідно їх правильно налаштувати. Для цього треба ретельно опрацювати документацію та провести налагодження, що займає багато часу. Виконувати кожен раз ці дії під час створення проєкту на React є контрпродуктивним, тому з’явилися інструменти автоматизації. Вони дозволяють швидко розпочати роботу: не треба витрачати час на пошук потрібних залежностей та їх конфігурування.

Існує кілька відомих середовищ розроблення додатків на React: create-react-app, create-next-app та vite. На основі графіків, побудованих за допомогою вебсервісу npmtrends.com (див. рис. 2.13), можна сформувати уявлення про співвідношення їхньої популярності.

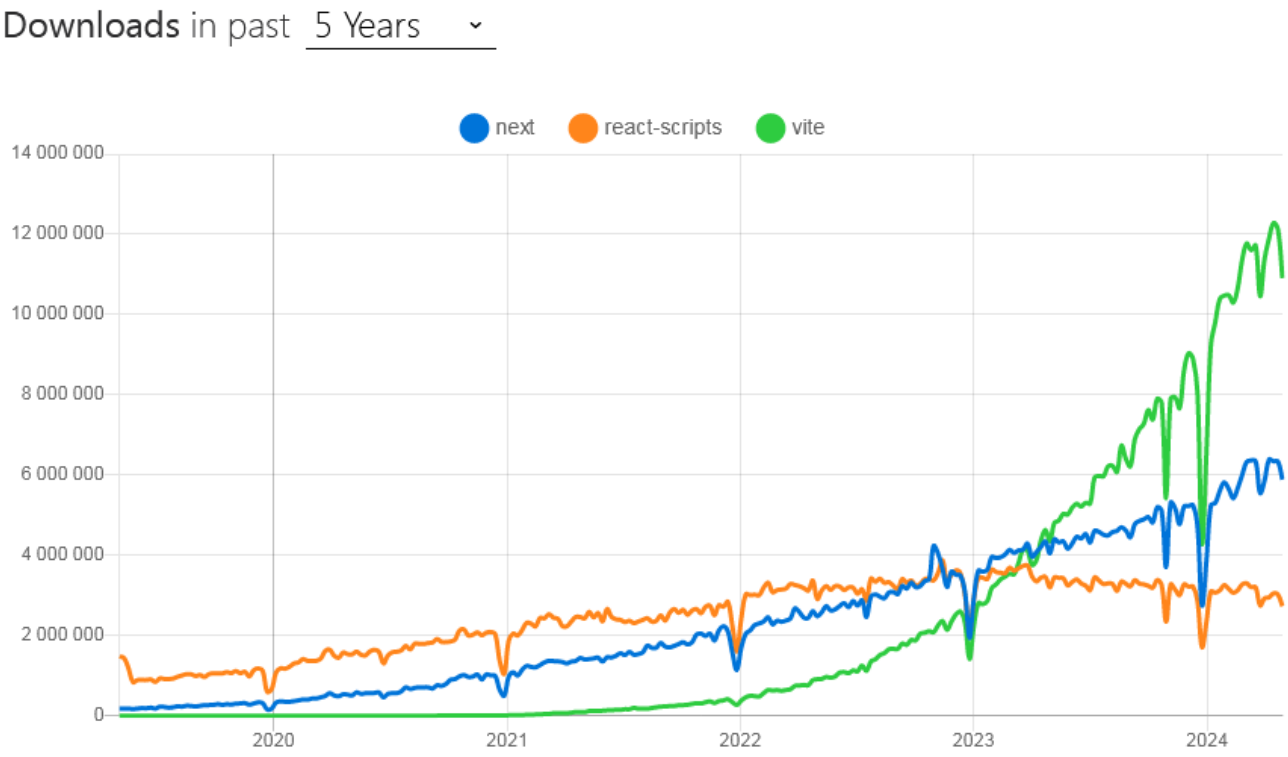


Рисунок 2.13 — Кількість завантажень інструментаріїв для розроблення додатків на React із репозиторію npm [71]

Примітка: проєкти, створені за допомогою create-react-app, містять залежність react-scripts, а ті, що розроблені з використанням create-next-app, базуються на next. Своєю чергою vite не пропонує окремих пакетів для керування.

Інструмент create-react-app є простим середовищем для розроблення додатків на React. Він підходить для початківців, оскільки пропонує нульову конфігурацію: налаштування транскомпілятора Babel та збиральника Webpack приховані. Наявний шаблон проєкту із підтримкою TypeScript.

Основним недоліком create-react-app є негнучкість: як максимум можна керувати параметрами компілятора TypeScript та пакета в цілому шляхом

редагування tsconfig.json та package.json відповідно. Щоб внести зміни до конфігурацій webpack або babel треба запустити сценарій «eject», що призведе до незворотних змін у проєкті, які позбавлять усіх переваг використання create-react-app.

Також виділяють проблему пошукової оптимізації (SEO), оскільки середовище розробки не має механізмів серверного рендерингу React-компонентів.

Найбільш суттєвим недоліком create-react-app є припинення підтримки цього середовища розробниками. Причиною тому став активний розвиток альтернатив, що мають ширші можливості та працюють значно швидше.

Більш популярним є create-next-app. Ключовою особливістю середовища стала підтримка серверного рендерингу. Також представлене вбудоване API для визначення маршрутів, що дозволяє імітувати переходи між сторінками в рамках додатку. Середовище базується на використанні зв’язки інструментів, Babel та Webpack. З версії 12 для покращення швидкодії запроваджується підтримка SWC за замовчуванням [72].

Середовище має свої недоліки. У першу чергу це неуніверсальність: для деяких проєктів запропоновані можливості з тісної взаємодії з сервером можуть виявитися надлишковими, оскільки працюємо не з React у чистому вигляді, а з фреймворком Next.js, побудованому на його основі. До того ж маємо є круту криву вивчення. Динамічна маршрутизація підтримується, утім повільніша за статичну [73].

Vite є найсучаснішим та найпопулярнішим середовищем розроблення застосунків на React. З метою пришвидшення запуску локального вебсервера використовується esbuild для попереднього збирання залежностей [74]. Для побудови production-збірок використовується Rollup, що хоч і повільніший за esbuild, але має потужну підтримку плагінів, що стало основою успішності екосистеми Vite [75]. Інструмент пропонує нульову конфігурацію, яка за необхідності може бути гнучко розширена. Наявна підтримка серверного рендерингу компонентів.

Середовище Vite базується на використанні переваг нативних модулів ECMAScript, що дозволяє заощадити час та генерувати збірки меншого розміру. Водночас маємо обмежену сумісність із різними браузерами, що є недоліком [76].

За результатами огляду середовищ, що використовуються для розроблення React-застосунків, обрано Vite. Для фреймворку зі створення віртуальних тренажерів надзвичайно важливо з плином часу залишатися актуальним, що вимагає застосування в його основі перспективних технологій.

Відповідно для транскомпіляції застосовуватиметься SWC. Побудовою production-збірки займатиметься Rollup. У якості локального вебсервера використовуватиметься Vite. Для роботи зі стилями обрано препроцесор SASS із варіантом синтаксису, що передбачає використання фігурних дужок (файли матимуть розширення «.scss»). Статичний аналіз забезпечуватимуть ESLint та Prettier.

Додатково встановлюватимуться плагіни, що містять правила статичного аналізу, спеціально розроблені для проєктів на React, а також ті, що дозволяють інтегрувати Prettier у робочий процес. Щоб запровадити типізацію для основних залежностей, треба встановлювати відповідні пакети з типами.

До того ж необхідна залежність для створення ZIP-архівів, що дозволить автоматизувати підготовку тренажерів до завантаження на сервер. Серед найбільш відомих: archiver, jszip та adm-zip. Перевагу надаємо першому. Він має більшу кількість завантажень у порівнянні з альтернативами (див. рис. 2.14), що пов’язано з активною підтримкою репозиторію.

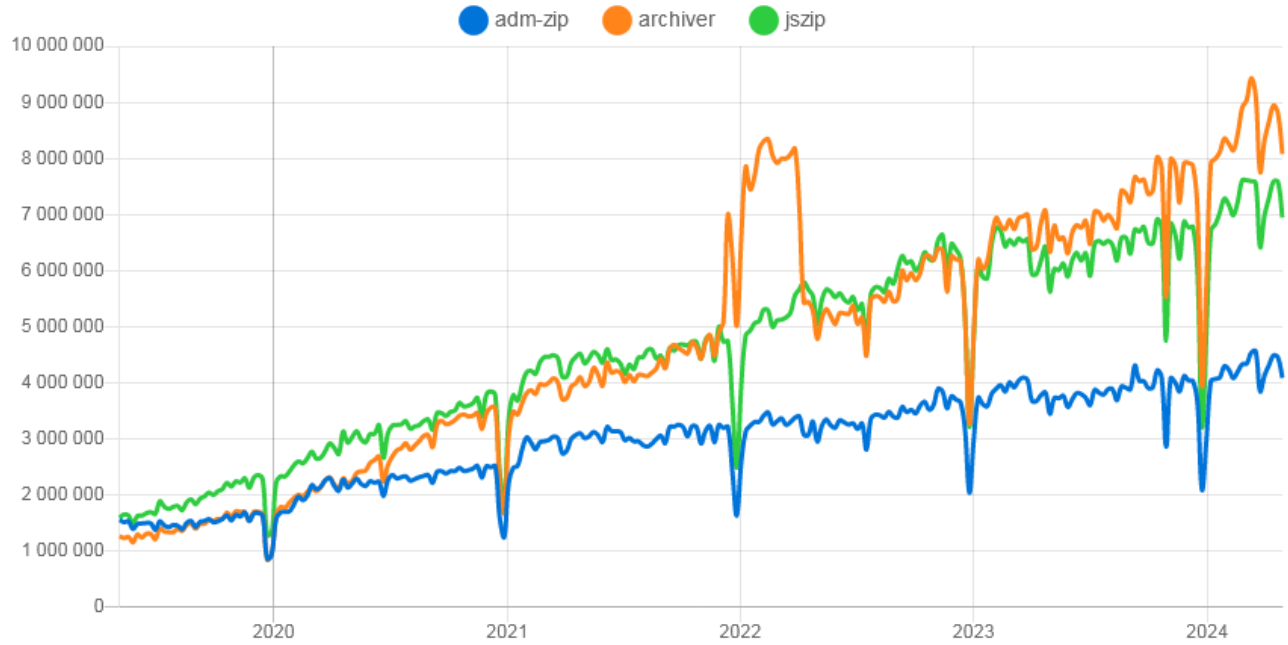


Рисунок 2.14 — Кількість завантажень пакетів для створення архівів ZIP із репозиторію npm [77]

Тренажери не потребують пошукової оптимізації, оскільки вони інтегруватимуться до навчальних платформ університету, де для доступу потрібна авторизація. Роботи не можуть проіндексувати такі ресурси.

В офіційній документації Vite зазначаються мінімальні версії основних браузерів: Chrome версії 87 і новіше, Firefox версії 78 і новіше, Safari версії 14 і новіше, Edge версії 88 і новіше [78]. Дати випуску цих версій: 17 листопада 2020 року для Chrome, 30 червня 2020 року для Firefox, 16 вересня 2020 року для Safari та 23 січня 2021 року для Edge за даними сервісу caniuse.com [79].

Рівень використання сучасних браузерів врахований під час обрання середовища. На сайті caniuse.com щомісяця публікується інформація стосовно їхніх часток на світовому ринку. Звіт являє собою зручну для опрацювання агрегацію даних, отриманих від StatCounter GlobalStats.

Результати аналізу звіту за березень 2024 року [80] подані в таблиці (див. табл. 2.2).

Таблиця 2.2 — Рівень підтримки з боку браузерів

Назва браузера	**Доля ринку**	**Рівень підтримки ESM**	**Рівень підтримки ESM у межах долі ринку**
Google Chrome	65.17%	64.42%	98.85%
Apple Safari	18.20%	17.46%	95.93%
Microsoft Edge	4.81%	4.77%	99.17%
Mozilla Firefox	2.70%	2.55%	94.44%
Усього	90.88%	89.20%	98.15%

Усі користувачі браузерів — це 100%. Вебпереглядачі відсортовані за спаданням долі ринку. Розмежування на мобільну та desktop-версію не проводиться, оскільки можливості браузера залежать від рушія, що є спільним для різних платформ. Значення в третьому стовпчику відображають відсоток користувачів версій браузерів із підтримкою нативних модулів. В останньому — та сама величина, але відносно долі ринку самого браузера.

Чотири браузери володіють долею ринку загальним розміром 90.88%. Середній рівень підтримки ESM серед використовуваних релізів перевищує 98%, що є відмінним показником. Це свідчить про те, що користувачі переважно оновлюють браузер, а тих, хто цього не робить, надзвичайно мало.

Решту ринку, 9.12%, ділять між собою велика кількість менш відомих альтернатив. Багато з них розробляються на основі проєкту Chromium, такі як Samsung Internet та Opera, що дає підстави робити припущення про них на основі аналізу можливостей Google Chrome.

Задокументовані мінімальні версії браузерів є вищими, аніж це вимагає використовуваний функціонал: нативні ES-модулі, динамічний імпорт та import.meta [81]. Причина: розробники Vite врахували виправлення дефектів, пов’язаних з цими аспектами, у різних версіях вебпереглядачів. Фактична підтримка є дещо ширшою.

У найгіршому випадку 89.20% усіх користувачів мають у себе достатньо сучасний браузер для успішного запуску React-додатків, створених за допомогою Vite, і цей показник поступово зростатиме. До того ж, для розробників передбачена ще одна опція з підвищення цього рівня: можна використовувати плагіни, як-от @vitejs/plugin-legacy, щоб забезпечити сумісність зі старими версіями браузерів.

Крім того варто враховувати особливості предметної області. Під час очного навчання системний адміністратор закладу освіти відповідає за оновлення браузерів на комп'ютерах в аудиторіях. Використання віртуальних тренажерів поширене в межах дисциплін ІТ-спеціальностей, студенти яких уміють самостійно підтримувати актуальність програмного забезпечення на своїх пристроях, що також є позитивним чинником.

Отже, обране середовище, Vite, відповідає часу.

Фреймворк для створення віртуальних тренажерів спирається не тільки на бібліотеку React, а й на інші основні залежності, завдяки чому отримуємо більш специфічний інструмент. Така адаптація до предметної області дозволяє більш ефективно розробляти навчальні об'єкти.

Реалізація механізму переходів між завданнями тренажера потребує додаткової бібліотеки керування станами. За допомогою вбудованого Context API хоч і можна це легко реалізувати, утім із численними недоліками, найбільш суттєвим серед яких є спричинення перемалювання компонента оновленням непов'язаного стану.

Бібліотека дозволяє цього уникати, що позитивно впливає на швидкодію вебдодатку. Код стає легше підтримувати, оскільки пропонується зрозумілий сталий підхід до роботи з даними. Відкривається доступ до перевикористання станів у різних частинах додатку. Тим самим знижуємо зв'язність компонентів, бо уникаємо антипатерну «prop drilling», коли передача властивостей відбувається через проміжні компоненти. Усі розглянуті переваги сприяють масштабованості продукту [82].

Серед бібліотек для керування станами є Redux, Recoil, Hookstate, Zustand, MobX, Rematch та Jotai [82]. За даними npmtrends.com, перша є лідером за кількістю завантажень (див. рис. 2.15).

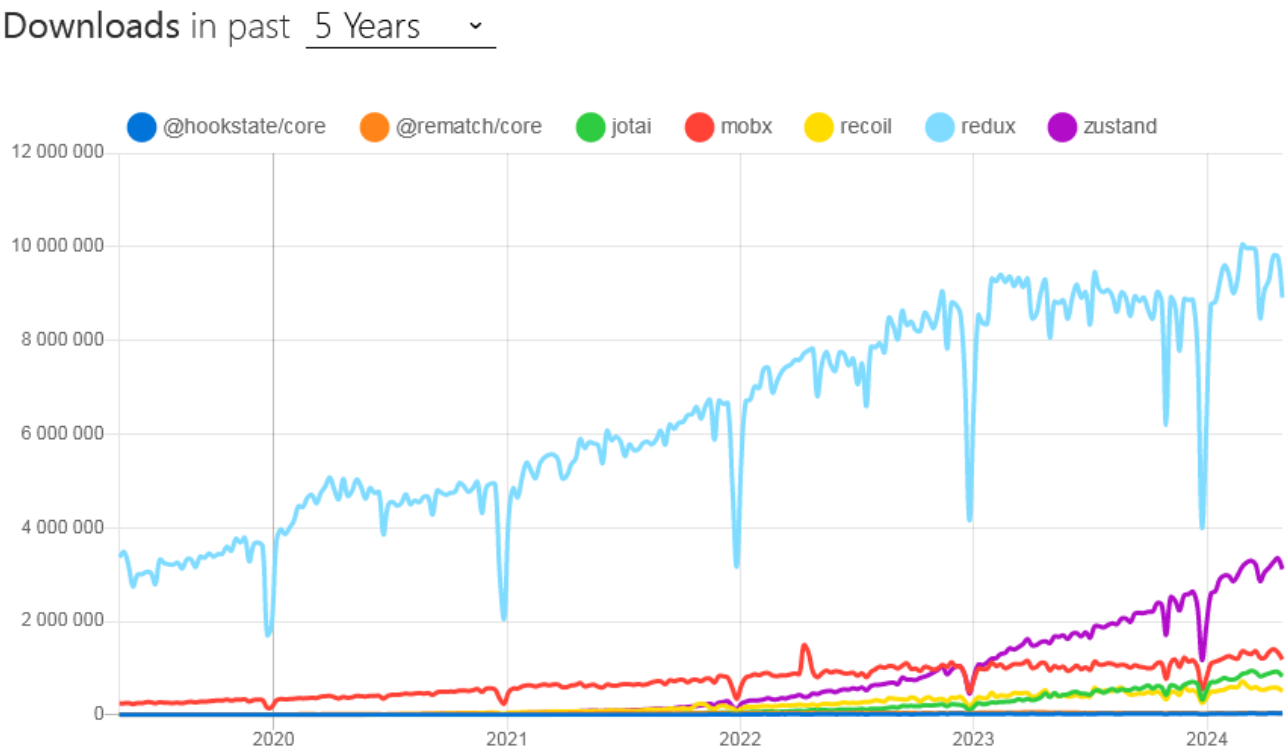


Рисунок 2.15 — Кількість завантажень бібліотек для керування станами із репозиторію npm [83]

Однією з переваг Redux є «single source of truth» — централізоване сховище станів. Його легко використовувати та підтримувати. Односпрямована модель потоку даних поліпшує налагодження та тестування додатку завдяки передбачуваності оновлень станів. Велика кількість супутніх бібліотек та інструментів з’явилася завдяки потужній спільноті. Серед недоліків: велика кількість шаблонного коду та складність концепцій для сприйняття початківцями [84].

Друге місце за популярністю стабільно тримає Zustand — легка бібліотека, що потребує мінімальних зусиль для впровадження. Підтримує визначення окремих сховищ станів, що дозволяє застосувати принцип розділення відповідальності з усіма випливаючими перевагами: легкість підтримки та тестування, а

також масштабованість. Звісно є й недоліки у вигляді обмеженої екосистеми через порівняно менший розмір спільноти та непристосованість до складних проєктів.

Інші рішення є менш популярними, як мінімум у 2 рази від Zustand. Вони мають свої переваги, утім для розроблюваного фреймворку важливо спиратися на достатньо популярний і водночас простий стек технологій, щоб забезпечити низький поріг входження для ширшого кола розробників навчальних об’єктів.

Для фреймворку зі створення віртуальних тренажерів більше підходить Zustand. Підтримка кількох сховищ корисна для відділення загальних станів тренажера від тих, що описують поточний етап розв’язання задачі (у нашому випадку транспортної). Крім того можна виокремлювати стани підсистем для дотримання належного рівня абстракції в архітектурі.

Zustand потребує мінімальної кількості коду для впровадження. Достатньо лише визначити сховище за допомогою функції create. За суттю воно є хуком, який потім використовується як селектор окремих станів або інструмент отримання всього вмісту сховища одразу за допомогою виклику getState. Redux-підхід виглядає складніше: після створення сховища потрібно визначити для нього компонент-провайдер, а також розробити дії та функції-редуктори для їхньої обробки.

Те, що екосистема, розвинута навколо Zustand, є меншою за аналогічну в Redux не є проблемою, оскільки користувачу фреймворку не потрібно буде розробляти механізм переходів між завданнями, бо надаватиметься вже готовий. Усе, що потрібно від розробника — описати вміст сховища станів вирішуваної задачі з використанням можливостей бібліотеки, а також створити об’єкти самих завдань.

Тренажер — це не велика складна система, а простий вбудовуваний додаток. Можливості бібліотеки Zustand у повній мірі задовольняють потреби розроблення без складної надлишковості.

Фреймворк надаватиме базові універсальні компоненти. За їх використання розробники зможуть зекономити час. До реалізації компонентів висувається кілька вимог.

По-перше, вони повинні бути кросбраузерними: тренажер має однаково відображатися в поширених версіях вебпереглядачів, щоб забезпечити якісний освітній процес.

По-друге, адаптивність: компоненти мають пристосовуватися до параметрів браузера користувача, зокрема розміру екрану, наявності підтримки функції наведення (hover) тощо.

По-третє, їхній вигляд має відповідати сучасним трендам дизайну та бути узгодженим між собою. Привабливий інтерфейс тренажера — важлива складова його ефективності в навчанні.

По-четверте, потрібно надати можливість кастомізації. Для застосування незначних змін не треба буде створювати окремий компонент із нуля.

За допомогою CSS фреймворків легко розробити власні компоненти, які б задовольняли зазначені вимоги. Інструментів досить багато і кожен дозволяє якісно виконати стилізацію, утім за різними підходами.

За результатами кількох опитувань «State of CSS» за рівнем використання Bootstrap є постійним лідером, а найближча конкуруюча альтернатива — TailwindCSS (див. рис. 2.16). Перевага на користь Bootstrap пояснюється не тільки його тривалою історією, а зручністю обраного рівня абстрагування, на якому працюють розробники. Значно простіше оперувати компонентами та стилями, аніж довгими комбінаціями низькорівневих атомарних класів-утиліт TailwindCSS. Використовуючи Bootstrap маємо змогу відділити контент від його стилізації.

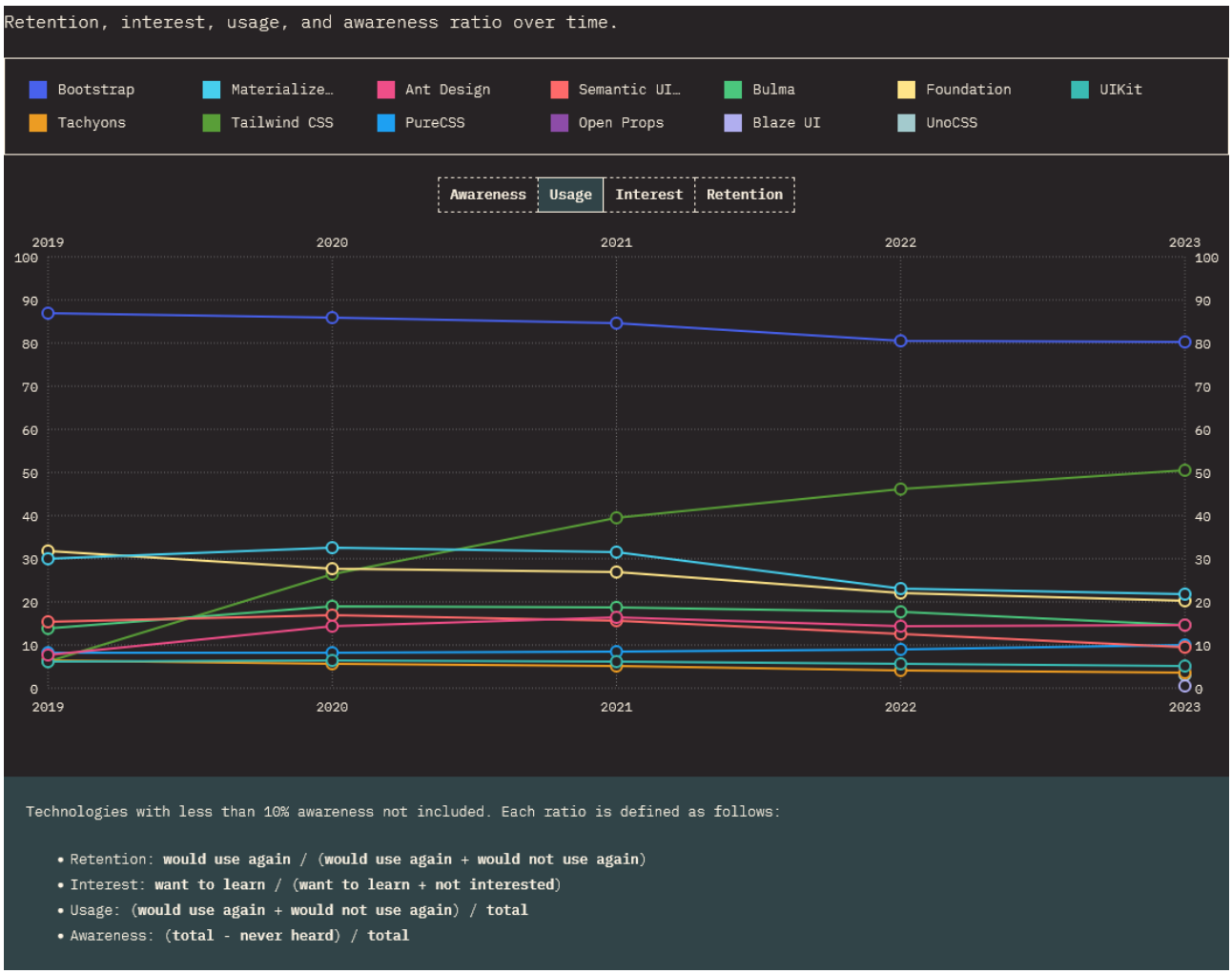


Рисунок 2.16 — Використання CSS фреймворків [85]

Додатково використовуватиметься Bootstrap Icons — бібліотека іконок. Вона містить спеціальні шрифти та CSS класи, за допомогою яких зручно додавати значки на вебсторінку. Обрана саме ця бібліотека з міркувань консистентності дизайну.

Відповідно фреймворк потребуватиме використання React Bootstrap, що є реалізацією Bootstrap-компонентів на основі React без додаткових залежностей [86]. Така адаптація дозволяє зручно інтегрувати компоненти CSS фреймворку до React-оточення. Це сприятиме підвищенню ефективності розроблення.

Анімація — важлива складова забезпечення інтерактивності навчальних додатків. З її допомогою можна звернути увагу на певні елементи вебсторінки. Крім того — це ефективний інструмент візуального зворотного зв’язку. У такий спосіб підвищується рівень залученості користувача до освітнього процесу.

У веброзробці існує декілька підходів до створення анімованого контенту.

CSS анімації — найбільш поширені, суть яких полягає в описі динамічних змін значень властивостей елементів за допомогою CSS. Вони легко налаштовуються та мають високу швидкодію, утім такий підхід надає дуже обмежений контроль над елементами DOM, що значно ускладнює створення складних анімацій.

JavaScript анімації базуються на керуванні DOM-елементами за допомогою сценаріїв. У такий спосіб отримуємо повний контроль, завдяки чому легко розробити послідовності перетворень. Серед недоліків: вимогливість до ресурсів та складність програмної реалізації.

SVG анімації — підхід, що спирається на можливості роботи браузерів із векторною графікою. Для опису анімацій використовується SMIL — мова синхронізованої інтеграції засобів мультимедіа на основі XML. Серед її можливостей: анімація числових, transform і колірних атрибутів елементів, а також рух за траєкторією [87]. Дозволяє точно керувати тривалістю анімацій, зациклювати їх, робити прив’язки до подій браузера та проводити синхронізацію. Серед недоліків: складний синтаксис та порівняно мала кількість ресурсів присвячених цій технології.

Canvas анімації працюють на основі сценаріїв JavaScript, що взаємодіють з елементами типу canvas (полотно) для виведення графіки. Створення зображень дуже гнучке завдяки піксельній обробці. Водночас маємо значну вимогливість до ресурсів пристрою. До того ж розробник має володіти різними алгоритмами для побудови зображень за пікселями.

Створюваний фреймворк використовуватиме переваги CSS анімацій для інтерактивізації віртуальних тренажерів. Невимогливість до апаратного забезпечення та легкість налаштування є ключовими перевагами обраного підходу. Обмеженість його можливостей не є проблемою: немає потреби у складних візуальних ефектах, оскільки вони більше відволікатимуть від навчального матеріалу, аніж розставлятимуть акценти.

З метою підтримки консистентності у межах проєкту та економії часу доречно скористатися однією з бібліотек, що містять готові до використання якісні анімації. Для цього обрано Animate.css, яка є найбільш популярною [88]. Крім комплекту готових анімацій бібліотека надає широкі можливості їхньої кастомізації. За допомогою класів-утиліт можна легко налаштувати тривалість, швидкість, кількість програвань тощо.

Для застосування анімацій до React-компонентів існує велика кількість бібліотек, зокрема React Spring, Framer Motion, React Transition Group, React Move та інші [89]. За кількістю завантажень React Transition Group є найбільш поширеною (див. рис. 2.17).

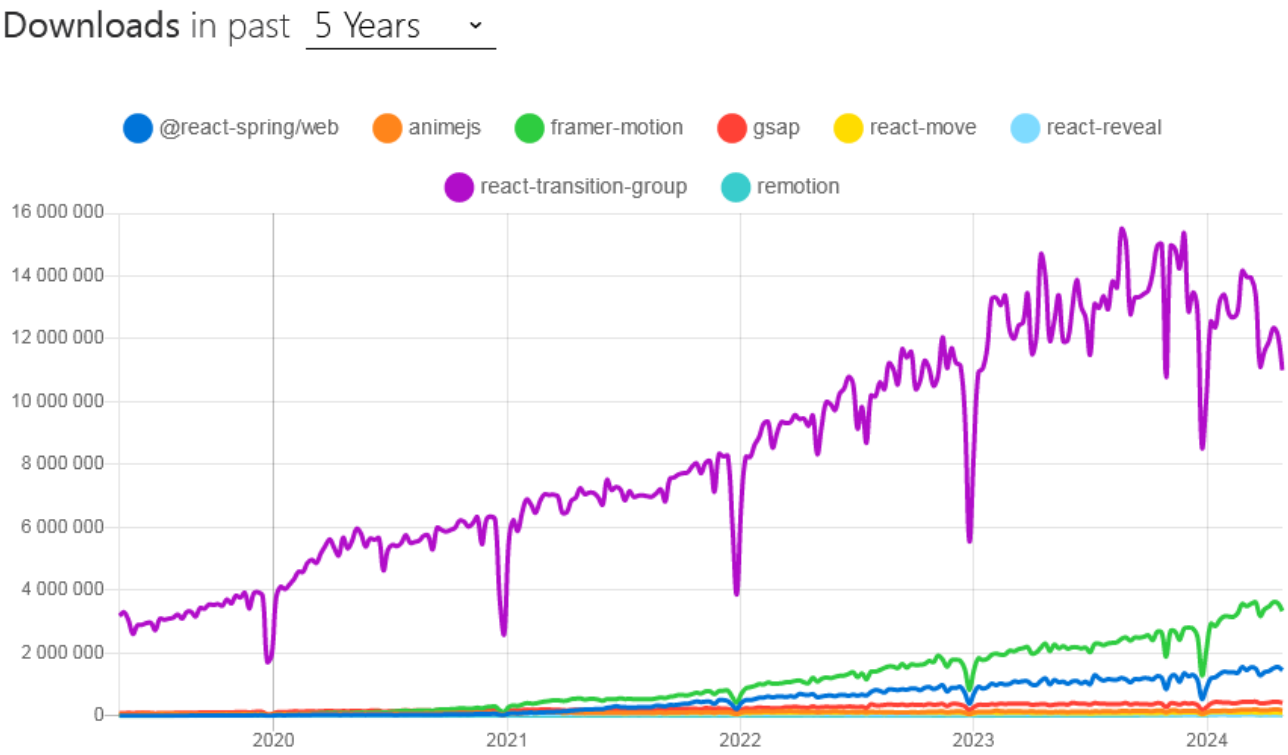


Рисунок 2.17 — Кількість завантажень бібліотек для керування анімаціями React-компонентів із репозиторію npm [90]

Бібліотека дозволяє керувати анімаціями компонентів шляхом додавання або видалення CSS-класів під час монтування та демонтування, а також у разі

оновлення станів. React Transition Group використовуватиметься в програмній реалізації запропонованої нової технології.

Під час розроблення досить часто виникає потреба в застосуванні класів до елементів за певних умов. Для формування значення атрибуту classes існує багато підходів. Серед найбільш популярних:

— побудова масиву назв, після чого викликається функція join із розділь-ником пробіл у якості аргументу;
— конкатенація рядків у циклі;
— виклик функцій Object.entries, filter, map, join;
— використання бібліотек.

Найбільш вдалим компромісом між швидкодією та читабельністю є підхід із використанням бібліотек. Серед найбільш відомих: classnames та clsx. За кількістю завантажень друга перехопила лідерство у вересні 2023 року й продовжує його утримувати (див. рис. 2.18).

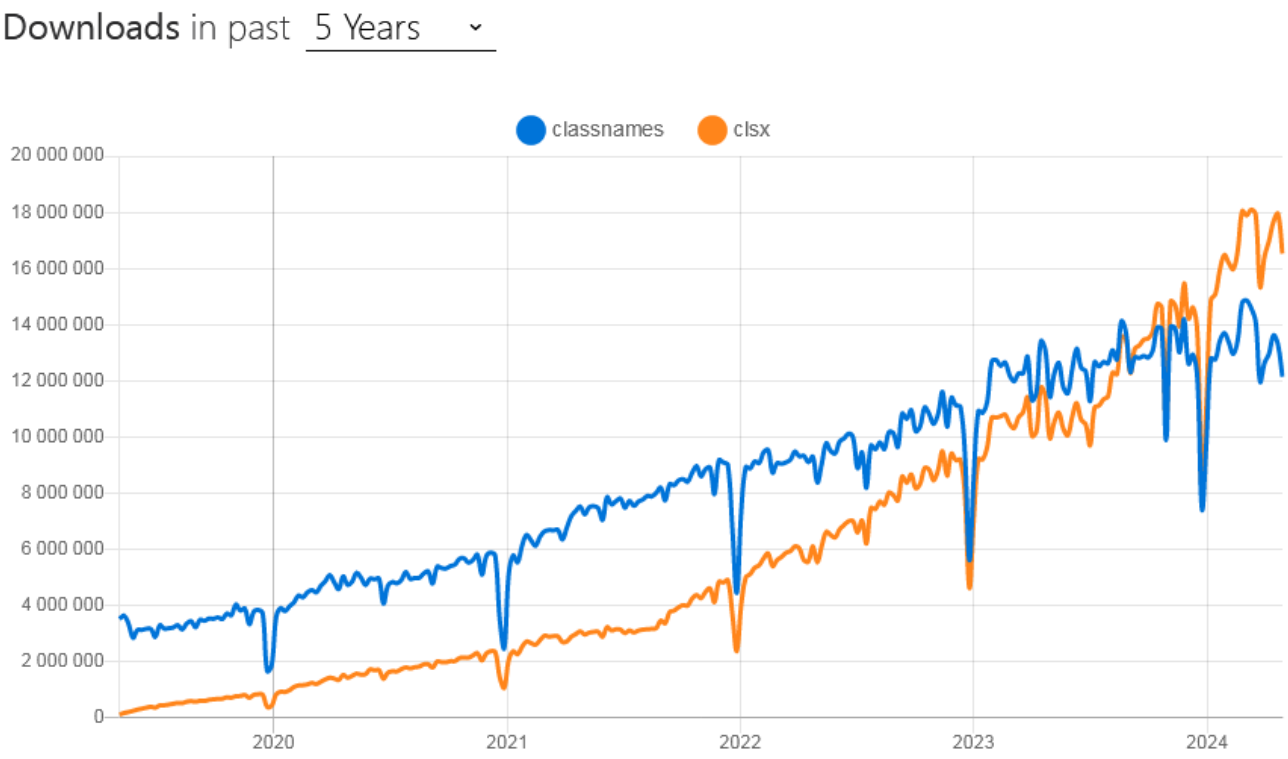


Рисунок 2.18 — Кількість завантажень бібліотек для застосування CSS-класів із репозиторію npm [91]

Бібліотека classnames постачається разом із альтернативними версіями: dedupe — для усунення дублікатів та bind — щоб працювати з класами за псевдонімами. Ці можливості практично не використовуються [92].

Своєю чергою clsx володіє вужчим функціоналом, але завдяки цьому займає менше місця та працює швидше за classnames, про що свідчать результати бенчмаркінгу (див. рис. 2.19), де критерієм оцінювання ефективності була кількість викликів на секунду.

Отже, для зручного опису правил застосування CSS класів прийнято рішення використовувати бібліотеку clsx.

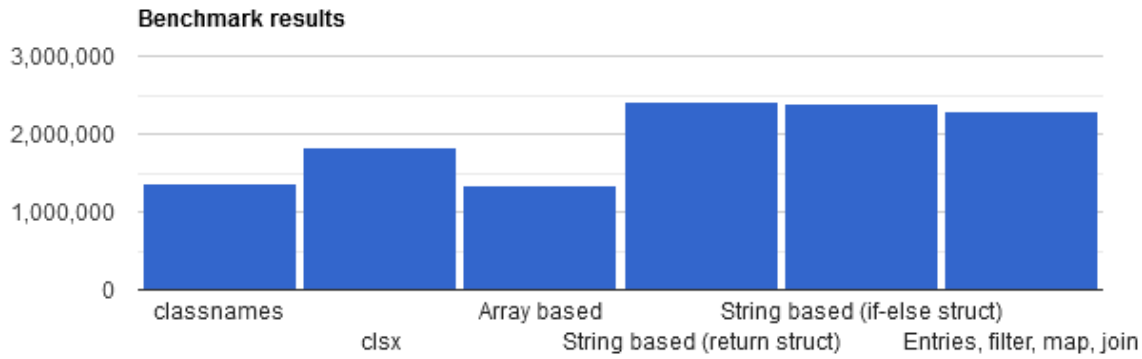


Рисунок 2.19 — Кількість операцій із отримання переліку CSS класів за різними підходами [93]

Специфікація ECMAScript не містить функцій для форматування дати за вказаним шаблоном, а така можливість потрібна для фреймворку. Наприклад, під час побудови звіту про результати виконання необхідно виводити час початку та завершення за форматом, що відповідає обраній користувачем локалізації. Створювати для цього власну функцію є контрпродуктивним: виникне проблема з розширюваністю — буде складно додавати підтримку інших мов.

Натомість це легко реалізувати за допомогою бібліотек для роботи з датами та часом, як-от date-fns та dayjs. Їхня популярність у колах розробників приблизно однакова, про що свідчить статистика завантажень (див. рис. 2.20).

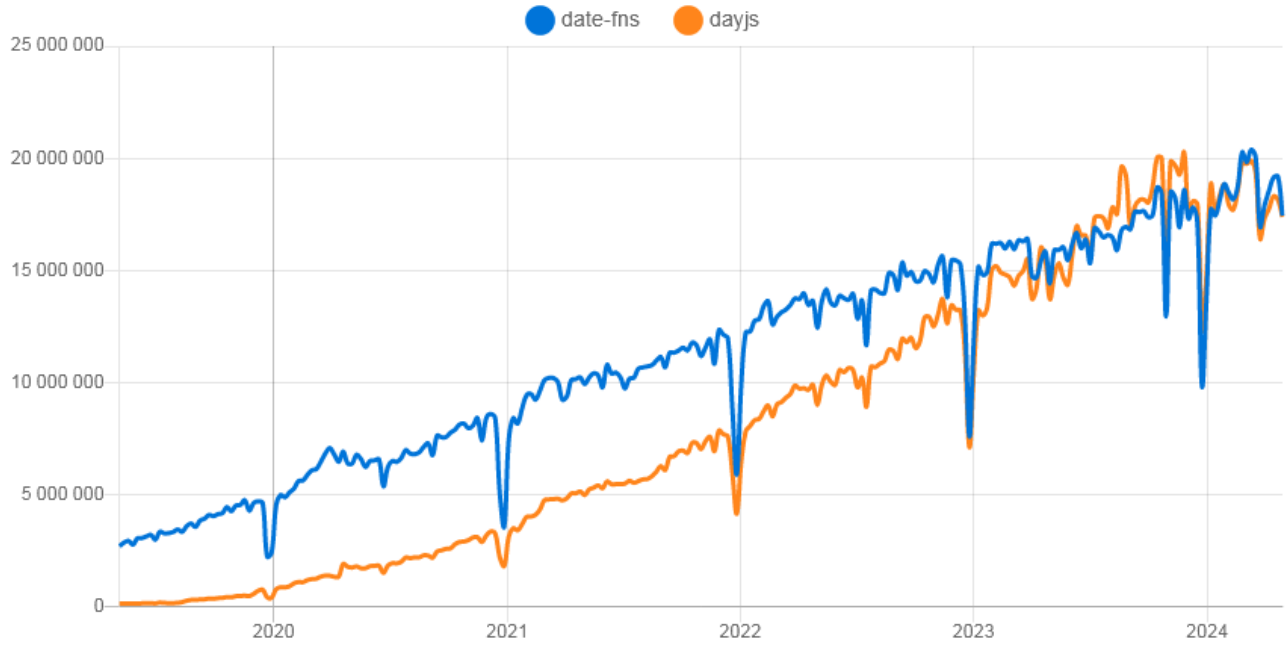


Рисунок 2.20 — Кількість завантажень бібліотек для роботи з часом із репозиторію npm [94]

Відмінностей небагато: date-fns трішки популярніший, часто працює швидше завдяки фокусу на оптимізації швидкості і має дещо менший розмір після збирання. Бібліотеки володіють схожим API, але пропонують різний доступ до нього: date-fns працює за функціональним підходом, а dayjs — на основі ланцюгових викликів [95].

Для використання у фреймворку надали перевагу date-fns з огляду на наявні переваги, хоч і незначні, над альтернативою.

Тренажери часто використовуються в рамках математичних дисциплін, тому фреймворк має включати в себе інструмент для відображення формул. Існує кілька способів їх візуалізації в браузері: MathML, растрові зображення та векторна графіка.

MathML — мова розмітки математичних формул на основі XML [96]. Не є дуже популярною через неоднозначну підтримку браузерами. Станом на 31 березня 2023 ситуація покращилася: рівень повної підтримки склав 94.35% від усіх

користувачів [97], утім альтернативні підходи встигли витіснити використання цієї технології.

Рівень підтримки зображень браузерами завжди був кращим, що стало підґрунтям до появи різних інструментів візуалізації формул. Рендеринг можна проводити як на боці сервера, так і клієнта — залежно від потреб проєкту.

Для JavaScript існує два популярні рушії візуалізації формул у нотації LaTeX: MathJax та KaTeX. Перший дозволяє відображати найрізноманітніші вирази. Функціональність другого вужча (підтримуються не всі вирази), але він працює швидше. Графіки кількості завантажень кожного (див. рис. 2.21).

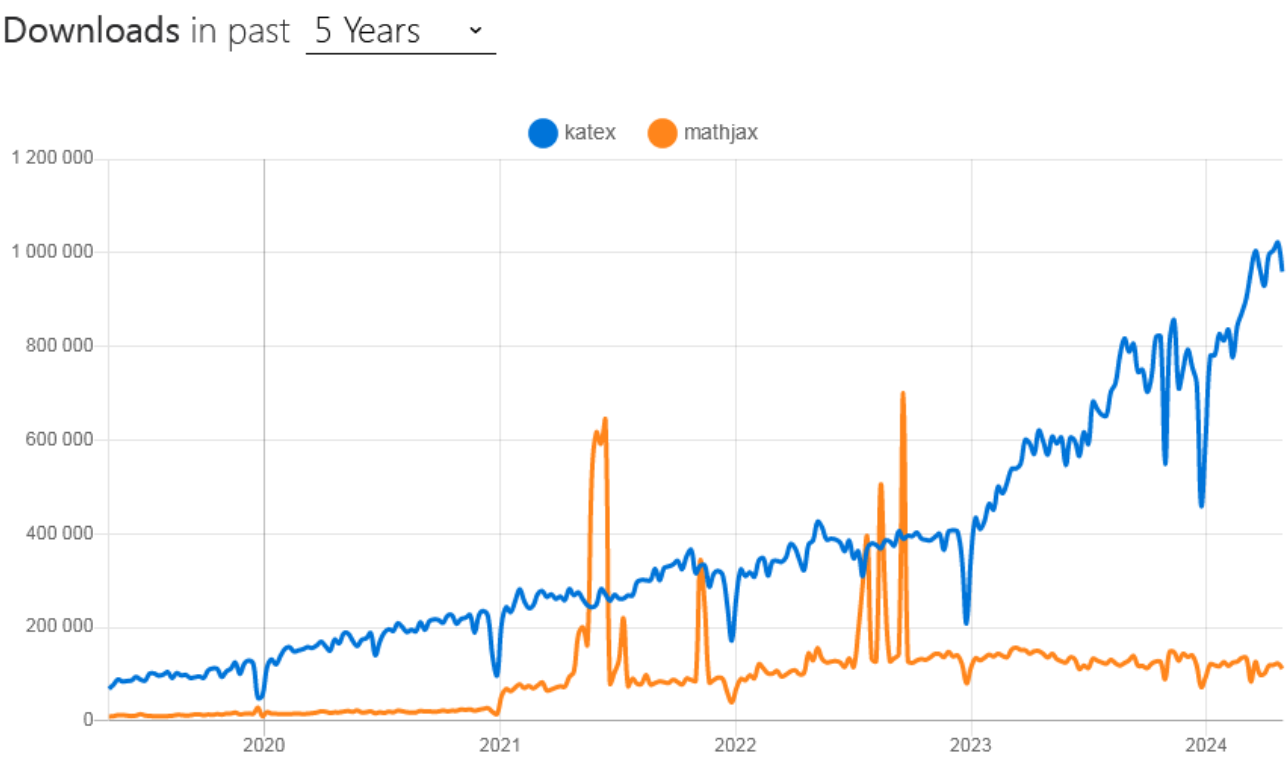


Рисунок 2.21 — Кількість завантажень бібліотек для роботи з LaTeX із репозиторію npm [98]

Для фреймворку зі створення тренажерів пріоритетним є повноцінна підтримка формул на LaTeX, тому надано перевагу MathJax. Обрана бібліотека не використовуватиметься самостійно, а в рамках хуку react-hook-mathjax, що

дозволяє зручно інтегрувати результати рендерингу математичних виразів до React-компонентів.

Підтримка функціоналу перетягування є важливою для забезпечення інтерактивності навчального об’єкта. У першу чергу — це про можливість гнучкої реорганізації інтерфейсу, що позитивно впливає на досвід користування.

Для цього в екосистемі React існує три відомих пакети: react-draggable, react-dnd та react-beautiful-dnd. Перший є дещо популярнішим за інші згідно зі статистикою, що надає сервіс npmtrends.com (див. рис. 2.22).

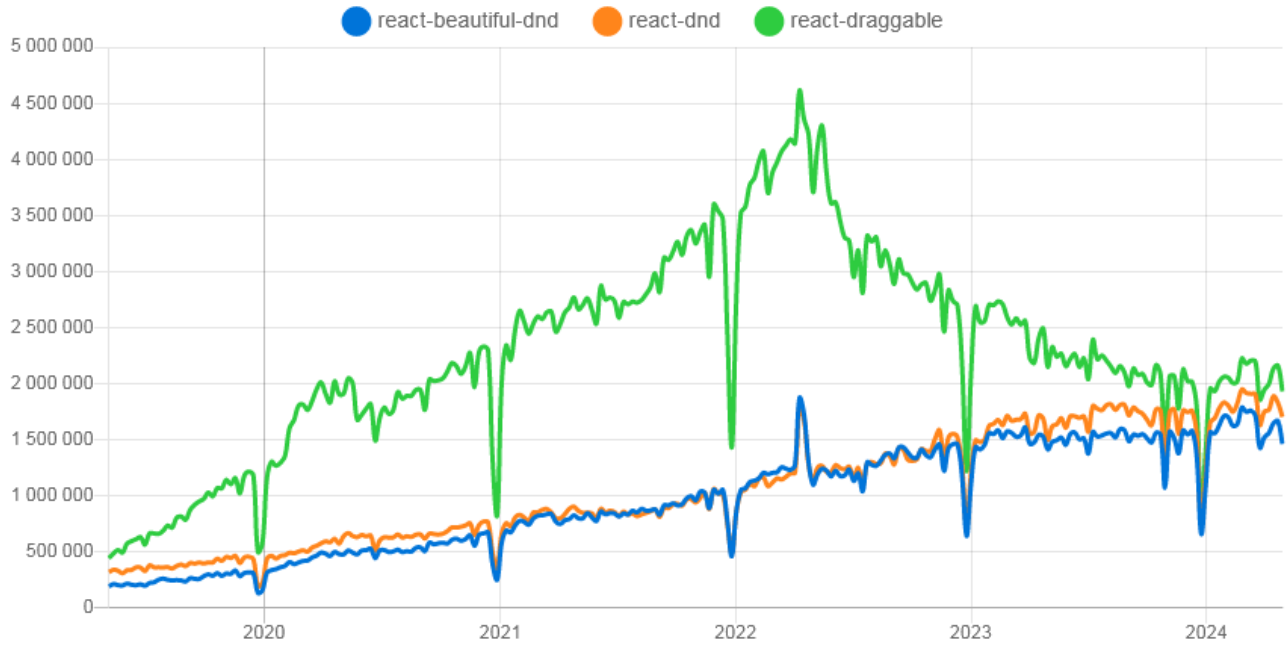


Рисунок 2.22 — Кількість завантажень із репозиторію npm бібліотек для підтримки перетягування [99]

Бібліотека react-draggable орієнтована на базові випадки перетягування, що не потребують складних налаштувань. Цих можливостей цілком достатньо для вбудованих додатків, таких як навчальні тренажери. Через це обрали саме цю бібліотеку.

Крім того фреймворк передбачає використання JSON Schema, за допомогою якої зручно формалізувати вимоги до вмісту JSON-файлів локалізацій. Для

використання переваг цієї технології, таких як валідація даних, потрібна підтримка з боку IDE.

Найбільш популярними опціями є Visual Studio Code та WebStorm.

Visual Studio Code — це безкоштовний легкий універсальний редактор коду від Microsoft, який можна перетворити на повноцінну IDE для веброзробки за допомогою плагінів, утім це може потребувати тривалого налаштування. Використовує простий механізм автодоповнення. Може бути надто вимогливим до ресурсів, якщо працювати з великим проєктом.

WebStorm — готове до роботи ергономічне середовище від JetBrains з усім необхідним для роботи. Пропонує потужні інструменти для рефакторингу коду. Володіє контекстним автодоповненням завдяки поглибленій підтримці мов веброзробки. Має покращену підтримку фреймворків і бібліотек. До того ж дозволяє ефективно працювати з великим проєктами.

Обрано WebStorm, оскільки за своєю суттю фреймворк для розроблення віртуальних тренажерів є достатньо великим за кодовою базою.

Загалом перелік технологій, із якими здатен працювати фреймворк, залишається відкритим: можна додавати залежності відповідно до потреб розроблення конкретного навчального об’єкта. Для створення тренажера з розв’язання транспортних задач можливості окресленого стеку технологій є вичерпними.

2.8 Методика освітнього експерименту

Метою освітнього експерименту «Ефективність застосування в освітньому процесі віртуальних навчальних тренажерів на основі автоматних моделей» була перевірка гіпотези кваліфікаційної роботи щодо підвищення ефективності віртуальних тренажерів в освітньому процесі завдяки використанню нової автоматної моделі з додатковою пам’яттю.

Ідея дослідження насамперед полягала в порівнянні навчальних досягнень та вражень, отриманих у результаті використання тренажерів на основі різних моделей двома вибірками, експериментальною та контрольною. Перша мала

працювати з тим, що розроблявся за допомогою запропонованого фреймворку (основним), а друга — з навчальним об’єктом на jTrainer (альтернативним). Очікувалося, що гнучка логіка формування завдань на основі автоматної моделі з додатковою пам’яттю забезпечить більшу повноту розкриття теми «Транспортні задачі», завдяки чому студенти матимуть кращий рівень знань у порівнянні з тими, хто займався з тренажером на більш простій моделі.

На етапі планування експерименту виділено кілька типів змінних: незалежні, залежні та контрольовані.

Серед незалежних змінних виділяють експериментальні та суб’єктивні. Серед перших те, чим можемо керувати. У нашому випадку це використовувана автоматна модель.

Незалежні змінні — чим можемо керувати під час проведення експерименту, досліджувані чинники впливу на значення залежних змінних. У нашому випадку це автоматна модель, на основі якої побудований використовуваний студентами тренажер. Залежною змінною виступає ефективність інтерактивного навчання.

Контрольовані змінні — умови проведення експерименту, сталість яких забезпечується для отримання точних результатів. Для проведеного експерименту ними є методика оцінювання, завдання поточного та підсумкового тестувань (повинні бути однакові для експериментальної та контрольної групи), структура анкети щодо отриманих вражень від роботи з тренажером, кількість навчальних годин, рівень попередньої підготовки, використовуваний навчальний матеріал тощо.

У якості бази для проведення дослідження обрана кафедра комп’ютерних наук Сумського державного університету. Заплановані обсяги експерименту:

— за розміром вибірок: до 50 студентів у кожній;

— за тривалістю: 3 тижні;

— за персоналом: здобувач освітнього ступеня магістр (автор роботи), викладач із дисципліни «Математичні методи дослідження операцій» (науковий керівник);
— за місцезнаходженням: дистанційно;
— за бюджетом: безкоштовно.

Учасники експерименту — студенти кафедри, що почали вивчати дисципліну «Математичні методи дослідження операцій». Збір даних до CSV-файлів та первинна обробка проведена із застосуванням табличного процесора Microsoft Excel 2016.

Формування контрольної та експериментальної групи відбувалося із використанням генератора псевдовипадкових чисел «Вихор Мерсенна» (MT19937) [100], що здатний забезпечити рівномірний розподіл. Прийнято рішення про призначення кожному студенту його групи (експериментальної чи контрольної) в індивідуальному порядку замість використання за основу навчальних груп. Це дозволило отримати більш загальні результати з нівелюванням впливу середньої успішності навчальних груп.

Підхід із застосуванням Microsoft Excel для поділу на дві однакові за розміром групи (або майже однакові, якщо загальна кількість учасників є непарною), досить простий. Кожному студенту треба поставити у відповідність випадкове число. Інтервал генерації має бути спільним для всіх. Для цього достатньо функції RAND, що повертає випадкові числа з інтервалу $[0;1)$. Застосовуючи функцію RANK.EQ можна отримати для кожного згенерованого значення його позицію у відсортованому в пам’яті за зростанням переліку. Щодо визначення приналежності: нехай експериментальній групі відповідає 1, а контрольній — 2. Загальна кількість учасників ділиться на 2, після чого застосовується округлення вгору, функція ROUNDUP, щоб врахувати випадок, коли загальна кількість учасників є непарною. У такий спосіб отримуємо розмір групи (більшої). Щоб перетворити позиції випадкового числа для кожного учасника необхідно виконати ділення на раніше знайдений максимальний розмір групи й застосувати

округлення вгору. Теоретично можливо, що такий підхід до поділу не спрацює, якщо для кількох учасників будуть згенеровані однакові випадкові числа, але алгоритм «Вихор Мерсенна» має величезний період повторення.

Електронна таблиця з первинними даними відповідає структурі (див. табл. 2.3).

Таблиця 2.3 — Структура електронної таблиці з даними про учасників

Стовпчик таблиці	**Призначення**
id	Ідентифікатор учасника
group_id	Ідентифікатор групи: 1 — експериментальна, 2 — контрольна
study_group	Назва групи, у якій навчається студент
student_number	Номер студента в рамках навчальної групи
full_name	Прізвище, ім’я, по-батькові
test_1_start_timestamp	Мітка часу початку поточного тестування
test_1_finish_timestamp	Мітка часу завершення поточного тестування
test_1_score_percentage	Оцінка у відсотках за виконання поточного тестування
trainer_last_finish_timestamp	Мітка часу останнього завершення роботи з тренажером
trainer_score_percentage	Максимальна оцінка у відсотках за виконання тренажера
test_2_start_timestamp	Мітка часу початку підсумкового тестування
test_2_finish_timestamp	Мітка часу завершення підсумкового тестування
test_2_score_percentage	Оцінка у відсотках за виконання підсумкового тестування
survey_finish_timestamp	Мітка часу завершення анкетування

Стовпці з особистими даними (study_group, student_number, full_name) потрібні лише на етапі компонування таблиці. Їхній уміст не оприлюднюється.

Для подальшої обробки даних обрано мову програмування Python. Для неї створено велику кількість бібліотек для роботи з даними, як-от pandas, numpy та matplotlib. Серед переваг використання Python-сценаріїв для проведення експерименту: економія часу завдяки автоматизації, гнучкість налаштувань процесу обробки й засобів візуалізації, забезпечення відтворюваності результатів та швидкість роботи з великими обсягами даних.

Під час статистичного аналізу з метою спростити опис логіки обчислень створювалися похідні колонки на основі первинних даних (див. табл. 2.4).

Таблиця 2.4 — Похідні дані

Стовпчик таблиці	**Призначення**
test_1_duration	Час виконання поточного тесту
test_2_duration	Час виконання підсумкового тесту
survey_answer_1	Відповідь учасника на перше запитання анкети
survey_answer_2	Відповідь учасника на друге запитання анкети
survey_answer_1_points	Оцінка відповіді учасника на перше запитання анкети
survey_answer_2_points	Оцінка відповіді учасника на друге запитання анкети
test_1_requirements_met	Результат перевірки виконання вимог до проходження поточного тестування
trainer_requirements_met	Результат перевірки виконання вимог до роботи з віртуальним тренажером
test_2_requirements_met	Результат перевірки виконання вимог до проходження підсумкового тестування
requirements_met	Результат перевірки виконання вимог до всіх етапів експерименту
duration_progress	Економія часу
score_percentage_progress	Приріст оцінки

Експеримент передбачав 5 етапів. На першому учасники мали опанувати матеріали курсу «Математичні методи дослідження операцій», такі як лекції та

практичні роботи. Після цього, на другому етапі, проводилося поточне тестування, щоб зібрати інформацію про початковий рівень знань студентів. Потім, на третьому, відкрився доступ до тренажерів. Учасникам експерименту зараховувалося проходження етапу за наявності щонайменше однієї успішної спроби, коли отримано 60% балів або більше. Під час четвертого студенти виконували підсумковий тест, що потрібно для визначення впливу від використання тренажерів. Завершальний, п’ятий етап експерименту, полягав в анкетуванні студентів для фіксації вражень від роботи із застосунками.

Конверсія розглядалася як цілком ймовірне явище: між етапами кількість учасників може зменшуватися, що пов’язано з тривалістю експерименту. Прийнято рішення про врахування даних лише тих студентів, що пройшли всі етапи. Формула для обчислення конверсії для етапу k:

$$C_k = \frac{n_k}{n_{k-1}} \cdot 100\% \tag{2.16}$$

де n_k — кількість учасників на поточному етапі;

n_{k-1} — кількість учасників на попередньому етапі.

Для організації експерименту підготований комплект матеріалів.

Поточний (див. додаток Б) та підсумковий тест (див. додаток В) складаються з 26 завдань кожний, що покривають основні етапи розв’язання транспортної задачі: визначення основних властивостей (4 запитання), отримання плану перевезень (11 запитань) та застосування методу потенціалів (11 запитань). Запитання сформульовані в межах основного матеріалу дисципліни «Математичні методи дослідження операцій». Особлива увага приділена варіативності: студенти в рамках одного тесту мали змогу попрацювати зі схожими ситуаціями, які мають певні відмінності. Це дозволило з’ясували, як учасники справляються з тонкощами процесу розв’язання. Наприклад, коли кількість запасів постачальника дорівнює обсягу потреб споживача і крок не є останнім — не можна одразу видаляти з розгляду обох учасників. Тести розроблені за схожим принципом,

щоб мати змогу коректно оцінити прогрес. Ризик запам’ятовування структури контролю знань усунутий налаштуванням випадкового порядку видачі завдань на платформі редактора навчальних матеріалів Lectur.ED. Підсумковий та поточний тест не містять ідентичних запитань.

Для збору зворотного зв’язку від студентів розроблено текст анкети (див. додаток Ж). Вона починається з преамбули, що розкриває суть експерименту. Для ідентифікації учасників використовувалося прізвище, ім’я, по-батькові, щоб уникнути врахування результатів сторонніх осіб. Запитання максимально прості: перше — про вплив на рівень мотивації, а друге — про повноту моделювання процесу розв’язання транспортної задачі. Під час розроблення варіантів відповідей для кожного використана шкала Лайкерта — психометрична шкала, яка вимагає від респондентів вказати ступінь згоди чи незгоди щодо об’єктів стимулу [101]. Обраний біполярний непарний, 5-бальний, різновид із середньою точкою.

У такий спосіб анкета являє собою перелік суджень по відношенню до тренажера. Висловлювання збалансовані: однаково представлені позитивні й негативні реакції на об’єкт. Відповіді кодуються знаковими цілими числами з однаковим кроком. Наприклад, у запитанні стосовно впливу на рівень мотивації: виключно негативний (−2), переважно негативний (−1), нейтральний (0), переважно позитивний (+1), виключно позитивний (+2). Відкриті запитання в кінці — лише для збору інформації про виявлені дефекти та нові ідеї для вдосконалення навчального об’єкта, не оцінюються. Прийнято рішення аналізувати результати в розрізі кожного запитання, а не на основі індексів, обчислених у вигляді зважених сум. У такий спосіб інтерпретація позбавлена суб’єктивності, що зазвичай вноситься використанням вагових коефіцієнтів.

До початку експерименту розроблений інструментарій й основний тренажер із його застосуванням. Альтернативний вебзастосунок створювався на основі jTrainer — фреймворку, на базі якого в Сумському державному університеті розроблятися інтерактивні навчальні матеріали після завершення підтримки технології Adobe Flash Player.

Технологія jTrainer не передбачає використання збиральників, бо кожен крок являє собою окремий файл-сценарій, що завантажується за необхідністю. Динамічність інтерфейсу реалізована за допомогою шаблонізатора mustache.js та бібліотеки jQuery. Для створення інтерфейсу додатку пропонується використання CSS фреймворку Bootstrap 3. jTrainer призначений для створення тренажерів із лінійною логікою виконання: без розгалужень та циклів.

Для забезпечення належної поінформованості студентів викладач дисципліни «Математичні методи дослідження операцій» попередньо проводив консультації щодо процедури участі в освітньому експерименті. Додатково розроблені супровідні матеріали: пам’ятка учасникам (див. додаток Г) й інструкція для експериментальної (див. додаток Д) та контрольної групи (див. додаток Е).

У якості програмного продукту для створення інфографіки обрано Adobe InDesign. Ключовою перевагою над альтернативами є можливість створення інтерактивних файлів PDF: підтримка налаштування посилань для зображень є нативним — немає потреби у виконанні маніпуляцій над прозорим текстом, як це реалізується в інших редакторах.

Пам’ятка учасникам освітнього експерименту (див. додаток Г) — дорожня карта з лаконічним описом етапів, ключовими датами та посиланнями на необхідні матеріали. Вона дозволила студентам краще зрозуміти процедуру дослідження та поліпшила навігацію матеріалами. Дана пам’ятка слугувала інструментом підвищення конверсії між етапами.

Інструкції до тренажерів — невеликі за обсягом PDF файли з відповідями на типові запитання щодо інтерфейсу користувача та умовних позначень. Ці матеріали також розроблені в середовищі Adobe InDesign (див. додатки Д, Е).

Чітко прописані вимоги до програмного забезпечення учасників експерименту. Мінімальні версії основних браузерів, що дозволяють запустити основний тренажер, узяті з документації Vite [78], наведені в таблиці (див. табл. 2.5).

Таблиця 2.5 — Мінімальні версії браузерів для основного тренажера

Браузер	**Мінімальна версія**	**Дата випуску**
Chrome	87	17.11.2020
Firefox	78	30.06.2020
Safari	14	16.09.2020
Edge	88	23.01.2021

Альтернативний тренажер має кращий рівень підтримки (97.4% станом на 18 квітня 2024 року за даними [102]), оскільки побудований на старих технологіях. Мінімальні вимоги визначалися виходячи з аналізу використаних можливостей CSS та JavaScript. На основі даних сайту caniuse.com встановлено: найбільш вимогливі аспекти реалізації відповідають специфікаціям CSS Flexible Box Layout Module та ECMAScript 5. Їх повноцінну підтримку забезпечують браузери, наведені в таблиці 2.6. Указані дати випуску стосуються desktop-версій.

Таблиця 2.6 — Мінімальні версії браузерів для альтернативного тренажера

Браузер	**Мінімальна версія**	**Дата випуску**
Chrome	23	25.09.2012
Firefox	28	18.03.2014
Safari	6.1	22.10.2013
Edge	12	29.07.2015

Перед використанням учасниками віртуальні тренажери мають перевірятися шляхом ручного тестування, щоб упевнитися в коректності запуску JavaScript сценаріїв та відповідності стильового оформлення між різними версіями браузерів.

Вимоги до операційної системи та апаратного забезпечення залежать від вебпереглядача, із яким працюватиме учасник експерименту. Основний на альтернативний віртуальний тренажер за рівнем навантаження на пристрій не

відрізняються від звичайних вебзастосунків. Припускаємо, що зазначені чинники не впливатимуть на результати, а тому не потребують деталізації. Єдиний виняток — вебкамера для участі в експерименті. Процедура проходження поточного та підсумкового тестувань передбачає дотримання принципів академічної доброчесності із застосуванням відеоконтролю. У такий спосіб знижується ризик спотворення результатів.

На платформі Lectur.ED створено два проєкти, кожен із яких містив пам’ятку, поточний тест, інструкцію до тренажера, тренажер, підсумковий тест та посилання на анкету. Це дозволило забезпечити роздільний доступ: учасники навчалися в ізольованих середовищах на Mіх. Завантажено по 5 версій пам’яток — для поетапного виведення посилань на пов’язані матеріали. Поступовість реалізована завдяки використанню механізму видалення/відновлення навчальних об’єктів. Копії анкети (див. додаток Ж) розміщені на Google Forms.

Під час проведення дослідження фіксувалися кількісні та якісні показники. Серед перших: бал у відсотках, час початку та завершення виконання тестів; а серед других: суб’єктивна оцінка рівня мотивації та повноти моделювання транспортної задачі за анкетою.

Очікувалося, що використання тренажерів у процесі навчання матиме позитивний статистично значущий вплив на кількість балів за тестування та часові витрати. Крім того основний тренажер мав продемонструвати вищу ефективність у порівнянні з альтернативним завдяки перевагам запропонованої нової автоматної моделі.

Гіпотеза кваліфікаційної роботи підтверджувалася шляхом виявлення статистично значущих переваг за різними показниками (успішність, час виконання завдань, рівень мотивації, повнота моделювання процесу розв’язання задачі тощо) на користь основного тренажера, із яким працювала експериментальна група.

Для перевірки гіпотез щодо нормальності розподілів результатів обрано критерій згоди Шапіро-Уілка, оскільки потрібно було працювати з невеликими

вибірками обсягом $n < 50$. За нульовою гіпотезою H_0 дані розподілені нормально, а за конкуруючою H_1 — за якимсь іншим законом.

Оскільки $n \in \overline{3,50}$, статистика критерію обчислювалася за формулою:

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_{(i)} \right)^2}{\sum_{i=1}^{n} \left(x_{(i)} - \overline{x} \right)^2}, \tag{2.17}$$

де a_i — коефіцієнт із SW-таблиць, залежить від розміру n вибірки;

$x_{(i)}$ — елемент відсортованого за зростанням варіаційного ряду;

$\overline{x}$ — середнє значення елементів досліджуваної вибірки.

Рівень значущості α визначає ймовірність помилки першого роду, коли відбувається відхилення істинної гіпотези H_0. Своєю чергою p — ймовірність отримати такі ж або більш екстремальні значення, за умови, що H_0 насправді є правильною.

Після обчислення W у таблицях критичних значень критерію відбувається пошук W_1, W_2 — таких значень, що знаходяться навколо W на рядку n, причому їхні рівні значущості $P_1 < \alpha$ та $P_2 > \alpha$ відповідно. Апроксимоване значення p обчислюється за формулою лінійної інтерполяції [103]:

$$p = P_1 + \frac{W - W_1}{W_2 - W_1} \cdot (P_2 - P_1). \tag{2.18}$$

У статистичних пакетах використовується розширення — критерій Ройстона Шапіро-Уілка, що додатково підтримує більші за розміром вибірки (до 5000) та має точнішу формулу обчислення p [104]:

$$\mu = 0.0038915 \cdot \left(\ln N \right)^3 - 0.083751 \cdot \left(\ln N \right)^2 - 0.31082 \cdot \ln N - 1.5861, \tag{2.19}$$

$$\sigma = e^{0.0030302\cdot(\ln N)^2 - 0.082676 - 0.4803}, \tag{2.20}$$

$$z = \frac{\ln(1-W) - \mu}{\sigma}, \tag{2.21}$$

$$p = 1 - \Phi(z), \tag{2.22}$$

де $\Phi(z)$ — інтегральна функція стандартного нормального розподілу:

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-\frac{t^2}{2}} dt \tag{2.23}$$

Умова прийняття гіпотези H_0: $p \geq \alpha$, а альтернативної H_1: $p < \alpha$. Якщо приймається H_0, тоді перевіряється рівність дисперсій із застосуванням параметричного F-критерію Фішера:

$$F = \frac{s_1^2}{s_2^2}, \tag{2.24}$$

де s_1^2 — більша вибіркова дисперсія;

s_2^2 — менша вибіркова дисперсія.

Основна гіпотеза H_0: дисперсії вибірок відрізняються несуттєво; альтернативна H_1: дисперсії значно відрізняються. Умова прийняття першої: $F \leq F_{кр}$. Критичне значення знаходиться з рівняння $P(F > F_{кр}) = \alpha / 2$, де α — рівень значущості.

Статистичні пакети використовують p замість критичних значень для перевірки гіпотез. Відповідно, умова прийняття основної гіпотези H_0: $p \geq \alpha$, а альтернативної H_1: $p < \alpha$.

Рівність дисперсій дозволяє застосувати параметричний t-критерій Стьюдента для двох незалежних вибірок:

$$t = \frac{\overline{x}_1 - \overline{x}_2}{\sqrt{\frac{s^2}{n_1} + \frac{s^2}{n_2}}}, \tag{2.25}$$

де $\overline{x}_1$, $\overline{x}_2$ — середні значення вибірок;

n_1, n_2 — розмір вибірок;

s — об’єднана дисперсія.

Остання обчислюється за формулою:

$$s^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}. \tag{2.26}$$

У такому разі кількість ступенів свободи знаходиться як:

$$df = n_1 + n_2 - 2. \tag{2.27}$$

Якщо рівність дисперсій попередньо відкинута, тоді t-критерій обчислюється за формулою:

$$t = \frac{\overline{x}_1 - \overline{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}, \tag{2.28}$$

де s_1^2, s_2^2 — вибіркові дисперсії.

Кількість ступенів свободи:

$$df = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{\left(s_1^2 / n_1\right)^2}{n_1 - 1} + \frac{\left(s_2^2 / n_2\right)^2}{n_2 - 1}}. \tag{2.29}$$

Правосторонній та лівосторонній тест Стьюдента використовуються для перевірки гіпотез про наявність значущих відмінностей між вибірковими середніми в одну сторону (більше або менше), а двосторонній — в обидві (наприклад, не дорівнюють). Їх властивості наведені в таблиці (див. табл. 2.7).

Таблиця 2.7 — Види тестів Стьюдента

Різновид t-тесту	**Нульова гіпотеза H_0**	**Альтернативна гіпотеза H_1**	**Критична область**
Правосторонній	$\overline{x}_1 \le \overline{x}_2$	$\overline{x}_1 > \overline{x}_2$	Справа: $P(t > t_{кр}) = \alpha$
Лівосторонній	$\overline{x}_1 \ge \overline{x}_2$	$\overline{x}_1 < \overline{x}_2$	Зліва: $P(t < t_{кр}) = \alpha$.
Двосторонній	$\overline{x}_1 = \overline{x}_2$	$\overline{x}_1 \ne \overline{x}_2$	З обох боків: $P(\lvert t \rvert < t_{кр}) = \frac{\alpha}{2}$

Якщо обчислена t-статистика потрапляє до критичної області, тоді нульова гіпотеза H_0 відкидається. У програмній реалізації достатньо оцінити обчислене значення p: значення, що дорівнює або перевищує α, підтверджуватиме основу гіпотезу H_0. В іншому випадку, $p < \alpha$, необхідно прийняти альтернативну гіпотезу H_1.

За допомогою критерію Стьюдента можна будувати довірчий інтервал CI для досліджуваної вибірки, щоб з обраним ступенем упевненості оцінити положення математичного сподівання μ в генеральній сукупності. Якщо дисперсія генеральної сукупності σ невідома, тоді знаходиться за формулою:

$$CI = \overline{x} \pm t_{\gamma} \frac{s}{\sqrt{n}}, \tag{2.30}$$

де $\overline{x}$ — вибіркове середнє;

γ — рівень довіри,

t_γ — значення за розподілом Стьюдента зі ступенями свободи $n-1$ та рівнем значущості $1-\gamma$;

s — вибіркове стандартне відхилення;

n — розмір вибірки.

Співвідношення $\frac{s}{\sqrt{n}}$ є помилкою репрезентативності.

У випадках, коли розподіл не є нормальним, передбачено застосування непараметричного U-критерію Манна-Уітні. Серед переваг подібних методів: не потребують конкретної моделі розподілу, стійкість до викидів та асиметрії даних. Вони застосовуються на невеликих обсягах даних.

U-критерій дозволяє перевіряти наявність статистично значущих відмінностей між двома вибірками, а саме приналежність до спільного закону розподілу. Критерій не порівнює статистики. Він враховує лише ранги значень вибірок в об'єднаному варіаційному ряді. Обчислюється за формулою:

$$U_i = R_i - \frac{n_i(n_i+1)}{2}, \tag{2.31}$$

де i — номер вибірки;

R_i — сума рангів значень i-вибірки;

n_i — обсяг i-вибірки.

Менше значення серед статистик U_i обираємо у якості тестового U. Якщо $U > U_{кр}$ (критичні значення можна знайти в таблиці), тоді приймається гіпотеза H_0, а якщо $U \le U_{кр}$ — альтернативна H_1.

Під час перевірки всіх статистичних гіпотез рівень значущості $\alpha = 0.05$. Точність показників: до трьох знаків після коми.

3 ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Програмна реалізація фреймворку

Верхній рівень структури проєкту тренажера на основі запропонованої технології-фреймворку (див. рис. 3.1) складається з конфігураційних файлів та двох директорій: src та public.

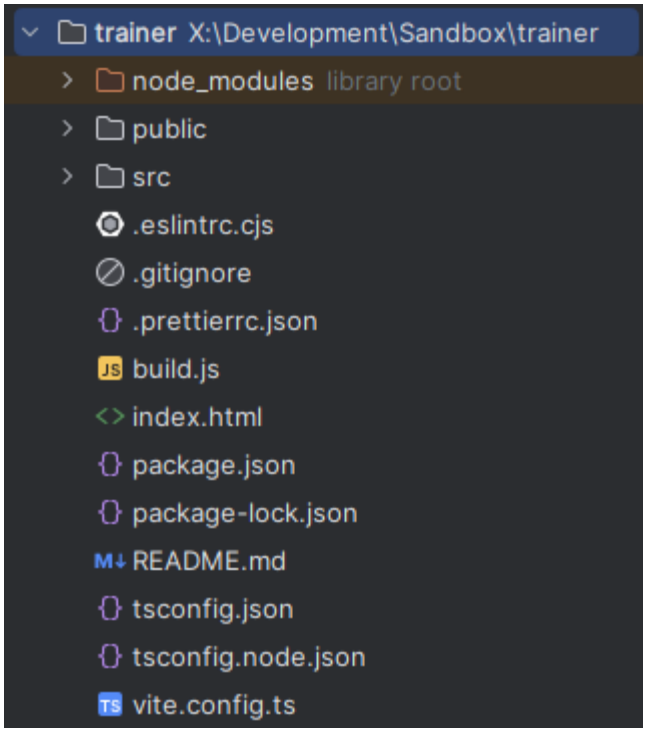


Рисунок 3.1 — Структура фреймворку для створення віртуальних тренажерів

У файлі package.json наведена основна інформація про пакет (див. рис. 3.2): назва (name), приватність (private), версія (version), тип модульної системи (type), скрипти керування проєктом (scripts), а також перелік runtime-залежностей (dependencies) і залежностей розроблення (devDependencies), обраних на етапі проєктування.

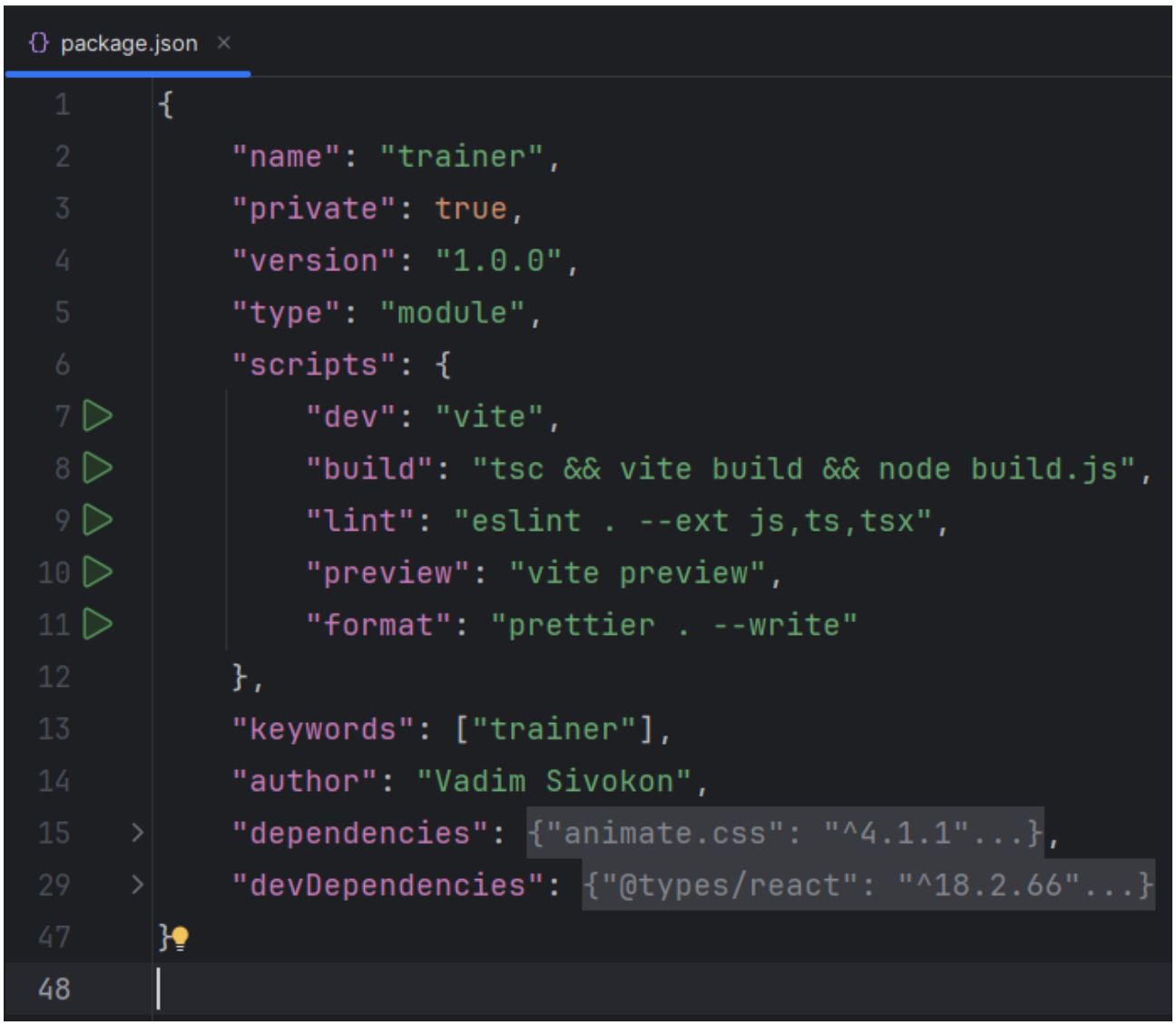


Рисунок 3.2 — Уміст package.json

За допомогою сценарію dev запускається локальний вебсервер із підтримкою «гарячої заміни модулів», що дозволяє слідкувати в реальному часі, як зміни в коді впливають на розроблюваний тренажер.

Виклик build запускає процес збирання проєкту, після чого передбачено виконання додаткової логіки, описаної в build.js (див. рис. 3.3), що адаптує збірку під вимоги Mix та Lectur.ED. Насамперед, ці сервіси очікують, що index.html — це не сам тренажер, а його iframe-обгортка (див. рис. 3.4). Для стиснення вмісту архіву застосовується підтримуваний сервером алгоритм Deflate з максимальним рівнем стиснення, що дорівнює 9.

build.js

```
import fs from "fs";
import path from "path";
import archiver from "archiver";

const packageJson = JSON.parse(fs.readFileSync( path: "package.json",  options: "utf8"));
const archiveName : string  = `${packageJson.name}_${packageJson.version}.zip`;
const outputDirectoryPath : string  = "./dist";

fs.renameSync(path.resolve(outputDirectoryPath, "index.html"), path.resolve(outputDirectoryPath, "trainer.html"));
fs.renameSync(path.resolve(outputDirectoryPath, "main.html"), path.resolve(outputDirectoryPath, "index.html"));

const output : WriteStream  = fs.createWriteStream(archiveName);
const archive : Archiver  = archiver( format: "zip",  options: {
    zlib: { level: 9 },
});
archive.pipe(output);
archive.directory(outputDirectoryPath,  destpath: false);
archive.finalize().then(() : void  => {
    fs.renameSync(archiveName, path.resolve(outputDirectoryPath, archiveName));
});
```

Рисунок 3.3 — Уміст build.js

index.html

```
<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <title>Trainer</title>
    </head>
    <body>
        <noscript>You need to enable JavaScript to run this app.</noscript>
        <div id="root"></div>
        <script type="module" src="/src/index.tsx"></script>
    </body>
</html>
```

Рисунок 3.4 — Уміст index.html

Команда lint дозволяє проводити лінтинг — статичний аналіз коду для виявлення потенційних помилок. Сценарій preview передбачає запуск локального вебсервера для перегляду production-версії сайту, яка має бути попередньо зібрана. Команда format необхідна для виправлення форматування коду в проєкті — у такий спосіб забезпечується консистентність стилю написання.

Файл .prettierrc.json містить налаштування для форматування коду, як-от ширина табуляції (tabWidth) у 4 символи та максимальна довжина рядка в 120 символів (див. рис. 3.5). Ці параметри змінені для поліпшення зручності перегляду вихідного коду фреймворку в редакторі коду.

Рисунок 3.5 — Налаштування Prettier

Налаштування ESLint розташовані у файлі .eslintrc.cjs (див. рис. 3.6). Конфігурація розширена врахуванням рекомендацій від Prettier на рівні попереджень. Вимкнено кілька суворих правил, що стосуються заборони появи змінних, які не використовуються («no-unused-vars») та застосування типу any («no-explicit-any»). Робота фреймворку базується на великій кількості callback-функцій (функцій зворотного виклику), для поліпшення сприйняття та розширення логіки яких простіше завжди наводити однаковий набір параметрів виклику. Щодо скасування другого правила: спрощуються описи типів узагальнених (generic) класів.

Рисунок 3.6 — Налаштування ESLint

Заповнений файл .gitignore (див. рис. 3.7) постачається разом із фреймворком, щоб спростити початок роботи над проєктом.

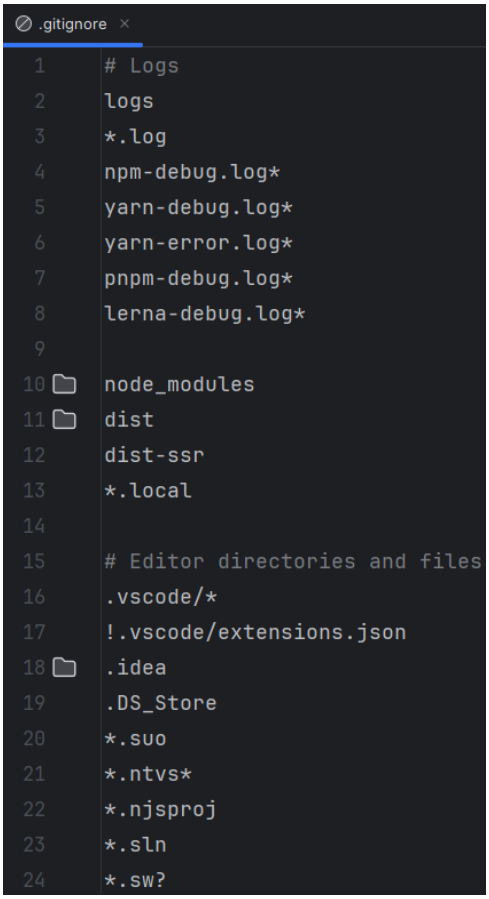


Рисунок 3.7 — Уміст .gitignore

У vite.config.ts наведені налаштування процесу розроблення та створення production-збірки (див. рис. 3.8). Оскільки тренажер — це вбудовуваний вебзастосунок, тому значення параметра base встановлюється рівним «./», що передбачено офіційною документацією для вбудованого розгортання (embedded deployment) [105]. За цільовий стандарт development (devTarget) та production-збірки (target) прийнято ECMAScript 2015, що дозволяє підвищити рівень сумісності зі старими версіями браузерів завдяки можливостям транспіляції SWC.

Рисунок 3.8 — Налаштування Vite

У файлах tsconfig.json та tsconfig.node.json містяться налаштування транскомпілятора TypeScript. Вони відповідають тим, що за замовчуванням пропонує Vite, за кількома винятками: змінено цільовий стандарт на ECMAScript 2015, а також вимкнені правила лінтингу стосовно змінних, що не використовуються.

Папка public призначена для збереження статичних ресурсів вебдодатку. Файли звідси ігноруються збиральниками. Доступ до цих ресурсів є прямим, бо їхні назви не змінюються. Файли main.html та report.html необхідні для коректної інтеграції до вебсторінок на платформах Mix та Lectur.ED. Перший містить опис елементу iframe — обгортки для контенту index.html, що в корені проєкту (цього вимагає Mix). Другий є порожнім файлом-заглушкою, наявність якої перевіряється під час завантаження тренажера на сервер університету.

У директорії src міститься основна частина програмної реалізації. Її структура (див. рис. 3.9).

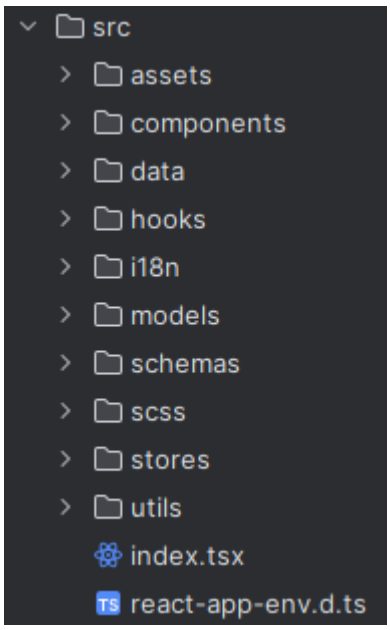


Рисунок 3.9 — Структура директорії src

Використання основних принципів ООП, SOLID, а також патернів проєктування забезпечили високу якість нової технології, яка відзначається своєю модульністю, розширюваністю та легкістю супроводу.

В основі архітектури тренажера на базі нової технології — кілька ієрархій об’єктів: завдання (Task), опції (Option), відповіді (Answer), види (React-компоненти), керувальні структури (Flowable), написи на дошці (Sketch), сповіщення (Alert), моделі оцінювання (GradingModel), дескриптори (Descriptor) та сутності предметної області.

На найвищому рівні абстракції визначений абстрактний клас ViewableModel та інтерфейси Flowable, Gradable та Cloneable.

Основною задачею абстрактного generic-класу ViewableModel (див. рис. 3.10) є зв’язування за допомогою фабричного методу createView класу моделі з її представленням у вигляді React-компонента. Крім того наявний функціонал для заміни заповнювачів конкретними значеннями або формульними вставками. Підстановки можуть бути локалізовані. Реалізовано механізм пререндерингу, суть якого полягає в попередній передачі даних до об’єкта моделі, тобто до оновлення представлення на вебсторінці. У такий спосіб пов’язані React-компоненти отримують константні дані безпосередньо від моделі з поля _props, а динамічні

— із поля _prerenderProps. Варто відзначити, що параметри-типи утворюють паралельні ієрархії.

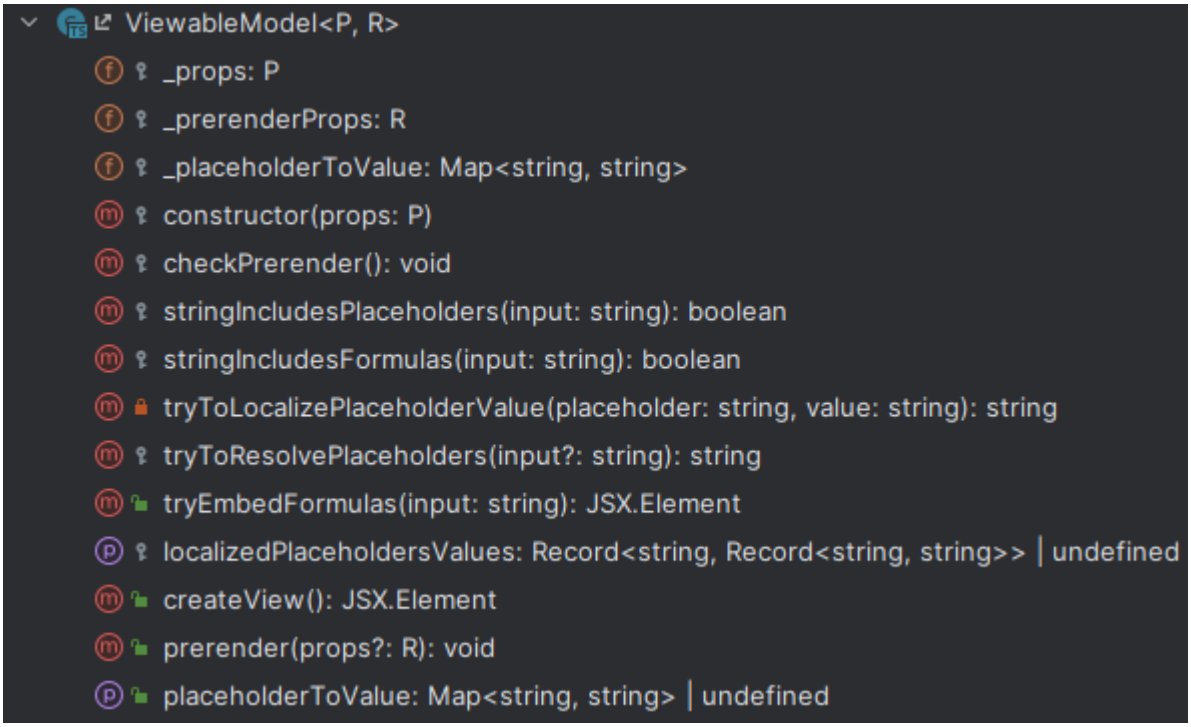


Рисунок 3.10 — Абстрактний клас ViewableModel

ViewableModel слугує основою для багатьох моделей, що мають графічне представлення, як-от завдання, опції, сповіщення та написи на дошці. Відповідно, усі можливості роботи із заповнювачами та формульними вставками підтримуються.

Інтерфейс Flowable (див. рис. 3.11) визначає методи, на основі яких відбувається обхід n-арного дерева з керувальними вузлами. Метод next повертає наступне завдання зсередини структури. Якщо його не існує — undefined, що означатиме необхідність передачі управління наступній структурі завдань. Абстрактний клас Task також є Flowable: під час першого виклику next він повертає this, а далі — виключно undefined. Метод reset необхідний для скидання стану структури завдань до початкового на випадок, якщо вона знаходиться всередині циклу. Класи керувальних вузлів, що реалізують Flowable, наведені в таблиці (див. табл. 3.1).

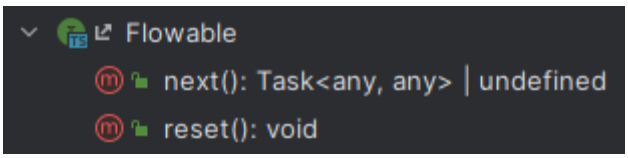


Рисунок 3.11 — Інтерфейс Flowable

Таблиця 3.1 — Класи керувальних вузлів

Клас	**Призначення**
FlowableAction	Проміжна дія потоку задач: виконується один раз, завжди повертає undefined у разі виклику методу next
LinearTaskFlow	Послідовна передача управління між вузлами дерева
ForLoopTaskFlow	Повторення фрагменту потоку завдань задану кількість разів
WhileLoopTaskFlow	Повторення фрагменту потоку завдань на основі передумови
LoopWhileTaskFlow	Повторення фрагменту потоку завдань на основі післяумови
SwitchTaskFlow	Обирає першу гілку, для якої справджується умова; якщо такої знайти не вдалося — керування передається далі

Логіка тренажера побудована на використанні функцій зворотного виклику. Для кожного завдання Task (див. рис. 3.12) можна налаштувати:

— validator — предикат, який приймає відповідь користувача та поточний стан Zustand сховища для роботи із потоком завдань;

— hasInfo — для виведення допоміжної інформації перед виконанням завдання, може бути постійним логічним значенням або предикатом — для динамічного виведення значення; підказка відображається лише один раз для одного й того ж завдання;

— placeholderHandler — функція-обробник заповнювачів;

— onInitialize — описує, що має відбутися перед початком виконання завдання; якщо завдання в середині циклу — запускатиметься знову на кожній ітерації;

— onComplete — спрацює в разі повернення функцією-валідатором підтвердження правильності відповіді користувача;

— onError — визначає, як опрацьовуватиметься помилка (тут можна формувати контекстні підказки).

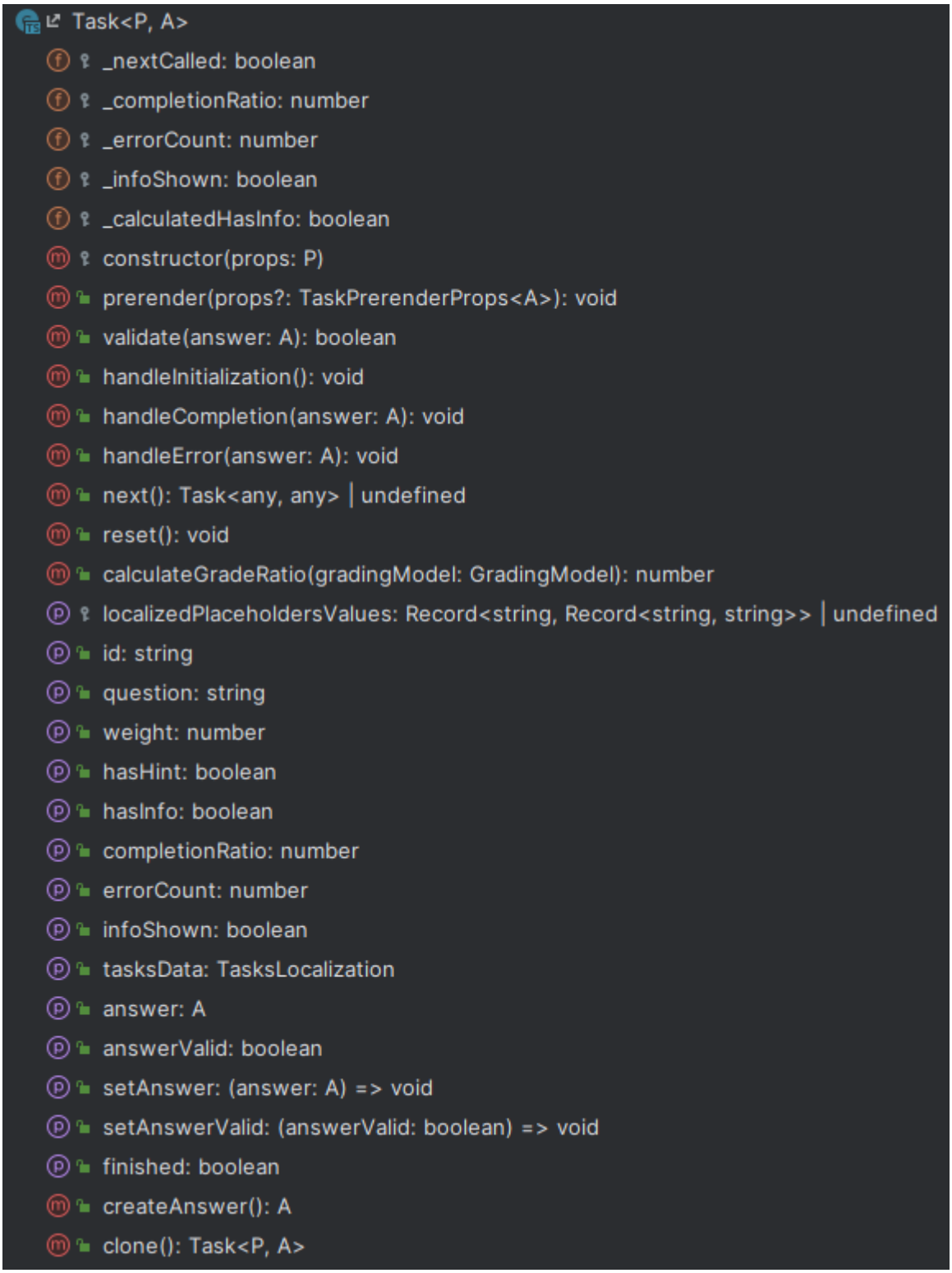


Рисунок 3.12 — Абстрактний клас Task

Передача значень між завданнями відбувається завдяки спільновикористовуваній змінній buffer. Вона являє собою практичну реалізацію механізму додаткової пам’яті автоматної моделі.

Дочірні класи для Task наведені в таблиці 3.2.

Таблиця 3.2 — Дочірні класи для Task

Клас	**Призначення**
FormulaCompletionTask	Доповнення формули цілим значенням
MouseSelectionTask	Обрання графічного елемента інтерфейсу
ChoiceTask	Абстрактний клас, дочірніми для якого є SingleChoiceTask (одна правильна відповідь) та MultipleChoiceTask (декілька правильних відповідей)

Підтримуються кілька видів опцій Option для завдань типу ChoiceTask (див. табл. 3.3).

Таблиця 3.3 — Класи опцій

Клас	**Призначення**
TextOption	Текстовий варіант відповіді
FormulaOption	Варіант відповіді у вигляді LaTeX-формули
TextAndFormulaOption	Варіант відповіді у вигляді LaTeX-формули, що супроводжується текстом

Для кожного типу завдання існує відповідний клас відповіді. У такий спосіб утворюється паралельна ієрархія, на верхньому рівні якої — абстрактний клас Answer. Таке архітектурне рішення обґрунтовується не тільки дотриманням принципу єдиної відповідальності, а й необхідністю оновлення графічного інтерфейсу, коли вміст сховища Zustand змінено. Базові класи відповідей наведені в таблиці (див. табл. 3.4). Зв’язок між класами завдання та відповіді встановлюється шляхом використання фабричного методу createAnswer.

Таблиця 3.4 — Базові класи відповідей

Клас	**Призначення**
FormulaCompletionAnswer	Зберігає відповідь користувача у вигляді рядка
MouseSelectionAnswer	Зберігає посилання на елемент графічного інтерфейсу із використанням дескрипторів
SingleChoiceAnswer	Зберігає щонайбільше один обраний користувачем варіант відповіді
MultipleChoiceAnswer	Може зберігати кілька обраних користувачем варіантів відповідей

Одним із ключових компонентів тренажера є віртуальна дошка, за допомогою якої можна виводити формалізований опис процесу розв’язання задачі в міру надання користувачем відповідей на запитання тренажера. Класи моделей, нащадки Sketch, наведені в таблиці (див. табл. 3.5).

Таблиця 3.5 — Класи написів на віртуальній дошці

Клас	**Призначення**
FormulaSketch	Формула
LabeledFormulaSketch	Формула з текстовим написом
HeadingSketch	Заголовок
ParagraphSketch	Абзац
SeparatorSketch	Роздільник
SpaceSketch	Відступ

У фреймворку передбачена підтримка 4 класів сповіщень — нащадків Alert (див. табл. 3.6). Вони відрізняються між собою стильовим оформленням.

Таблиця 3.6 — Класи сповіщень

Клас	**Призначення**
InfoAlert	Інформаційне повідомлення
SuccessAlert	Повідомлення про успіх

Закінчення таблиці 3.6

Клас	**Призначення**
WarningAlert	Попередження
ErrorAlert	Повідомлення про помилку

Моделі оцінювання визначаються на основі абстрактного класу Grading-Model (див. рис. 3.13).

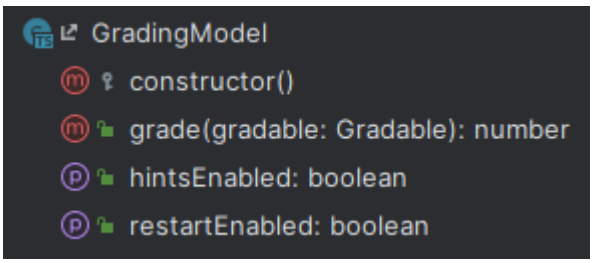


Рисунок 3.13 — Структура класу GradingModel

Модель визначає не тільки алгоритм обчислення кількості балів, а й доступність різних можливостей тренажера, таких як його перезапуск та формування контекстних підказок після надання неправильних відповідей.

Модель оцінювання може працювати з об’єктами, класи яких реалізують інтерфейс Gradable (див. рис. 3.14). Flowable є розширенням цього інтерфейсу, тому всі типи завдань та їх структури є оцінюваними.

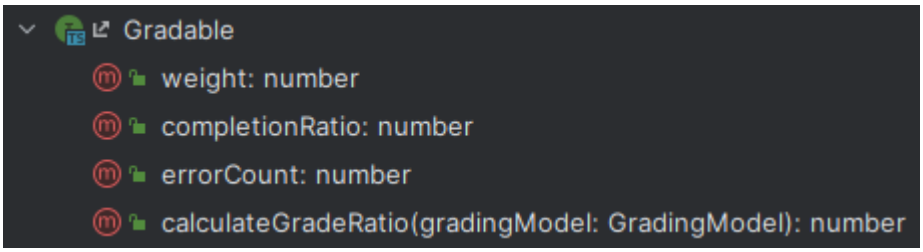


Рисунок 3.14 — Інтерфейс Gradable

Кожен об’єкт, клас якого реалізує Gradable, може відслідковувати кількість помилок користувача на поточній ітерації. Крім того мають бути реалізовані методи weight та completionRatio, що повертатимуть вагу завдання в межах

батьківської структури та долю виконання відповідно. Ця інформація опрацьовується моделлю оцінювання.

Кожен клас, об’єкти якого є оцінюваними, має свої особливості. До прикладу, будь-яке завдання повертає 1 у якості completionRatio, якщо отримана правильна відповідь, та 0, якщо ні. Проміжні дії, об’єкти класу FlowableAction, мають нульову вагу. Оцінка за виконання циклу завдань зараховується тільки після повного виходу зі всіх циклів. Розгалуження, SwitchTaskFlow, вважається успішно виконаним навіть тоді, коли жодна з гілок не обрана.

Реалізовані базові дві моделі оцінювання: TrainingEvaluationModel та KnowledgeEvaluationModel. Перша — менш сувора: дозволяє перезапускати тренажер та відображати користувачу контекстні підказки, кількість помилок не впливає на оцінку. Друга — це забороняє й, крім того, максимальний бал зменшується до 50% за наявності хоча б однієї помилки.

Дескриптор — об’єкт, на основі даних якого відбувається пошук іншого об’єкта. Функціонування тренажера за новою технологією передбачає використання дескрипторів у якості посилань на елементи графічного інтерфейсу, що є важливою складовою формування контекстних підказок для користувача. Ієрархія будується відносно абстрактного класу Descriptor. На початку роботи з фреймворком конкретною реалізацією моделі є тільки SketchDescriptor.

На основі інтерфейсу Cloneable реалізований зручний механізм клонування об’єктів. Інтерфейс реалізовується тими класами, об’єкти яких можуть використовуватися в станах React-застосунку. У такий спосіб логіка створення точної глибокої копії описується в методі clone, що позбавляє необхідності у використанні додаткових бібліотек для цього. Процедура оновлення стану максимально проста: створити об’єкт-клон для наявного стану, провести зміни необхідних полів, після чого передати цей об’єкт до функції-мутатора.

Папка assets призначена для збереження статичних ресурсів, як-от шрифти та зображення. Для стилізації результату калькулятора використаний шрифт «Digital Numbers» Стефана Альфа, а для написів на віртуальній дошці — «LC

Chalk» від Lazy Crazy. Зображення прапорів локалізацій створені за допомогою Adobe Illustrator — редактора векторної графіки. Представлена атрибутика Сумського державного університету у вигляді логотипів.

У папці components містяться представлення базових універсальних компонентів, які можна перевикористовувати:

- віртуальна дошка та її елементи (формула, формула з написом, заголовок, абзац, роздільник, відступ), що дозволяють поступово виводити формалізований опис процесу розв’язання задачі;
- цифровий годинник (DigitalClock), який демонструє поточні витрати часу на виконання завдань;
- калькулятор (Calculator) із підтримкою виконання кількох операцій одночасно;
- шкала прогресу (TrainerProgress), що візуалізує поточний прогрес проходження тренажера;
- картка завдання (TaskCard) та базові типи запитань:
  - доповнення формули цілим числом (FormulaCompletionTaskView);
  - обрання елемента графічного інтерфейсу за допомогою миші (MouseSelectionTaskView);
  - обрання однієї або кількох правильних відповідей (ChoiceTaskView) із підтримкою варіантів відповідей у вигляді текстового напису (TextOptionView), формули (FormulaOptionView) та комбінованого (TextAndFormulaOptionView);
- селектор режиму роботи (ModeSelector);
- універсальний ефект блимання (Blinking) для привернення уваги до певних елементів графічного інтерфейсу;
- діалогові вікна підтвердження дії (ConfirmationDialog), введення даних (InputDialog) та перегляду додаткової інформації (InfoDialog);
- сповіщення (AlertView) та їх перелік (Alerts) для побудови зворотного зв’язку з користувачем у відповідь на його дії;

— селектор мови (LanguageSelector), за допомогою якого можна швидко перемикатися між локалізаціями;
— цілочисельне поле введення (IntegerInput), для якого налаштована велика кількість перевірок, щоб унеможливити потрапляння некоректних даних до тренажера;
— основні фрагменти макету сторінки — основний контент (Main), верхній (Header) та нижній колонтитул (Footer); перший описує взаємне розташування базових компонентів між собою, коли як колонтитули — верхню (логотип, назва тренажера, селектор мови) та нижню частину сторінки (кнопки виклику калькулятора, перезавантаження та завершення тренажера) відповідно.

Крім того є спеціальний компонент для передзавантаження ресурсів — Preloader. Насамперед його потребує віртуальна дошка для уникнення помилок рендерингу LaTeX-формул, які можуть виникати через несвоєчасне підвантаження динамічних бібліотек MathJax. Логіка використання компонента: треба розмістити всередині нього формулу або інший статичний ресурс, що має завантажитися заздалегідь. Сам компонент не відображатиметься для користувача.

Директорія data призначена для розміщення вхідних даних, із якими працюватиме тренажер: основні налаштування (configuration.ts), підтримувані локалізації (i18n.ts), потік завдань (task-flow.ts) та його елементи в директорії tasks.

Конфігурація є централізованою. Вона дозволяє:

— керувати анімованістю всіх компонентів у тренажері;
— умикати/вимикати режим налагодження;
— вказувати значення за замовчуванням (наприклад, локалізацію);
— визначати спеціальні символи та позначки;
— гнучко налаштовувати виведення LaTeX-формул;
— конфігурувати вікна, що спливають;
— задавати тривалість відображення сповіщень різного типу (інформація, успіх, попередження та помилка);

— задавати параметри API клієнтської та серверної частини.

До того ж вона може бути використана для визначення глобальних змінних, що будуть видимі в будь-якій частині застосунку.

Папка hooks призначена для виокремлення повторюваної логіки до окремих хуків, що використовуватимуться в різних компонентах.

У директорії i18n зберігаються дані локалізацій в окремих піддиректоріях: ui, tasks та sketches. Перша відповідає за переклад тексту на основних елементах графічного інтерфейсу; друга — за локалізацію умов завдань та варіантів відповідей, а третя — для локалізації написів на віртуальній дошці. Дані локалізацій наведені у форматі JSON. Назви файлів локалізацій відповідають кодам мов згідно зі стандартом ISO 639. Вимоги до вмісту формалізуються шляхом застосування JSON Scheme. Вони розташовані у файлах ui.schema.json, tasks.schema.json та sketches.schema.json. У процесі розширення проєкту схеми варто доповнювати вручну.

Механізм зміни локалізацій реалізований у вигляді React-стану, завдяки чому перемикання відбувається миттєво. Можливість доступна на будь-якому етапі роботи з тренажером.

У папці stores зберігаються описи Zustand сховищ станів у вигляді хуків: useAppStore, useTaskStore та useCalculatorStore. Вони є доступними для всіх компонентів.

Стани, визначені в рамках app-store.ts, зберігають поточну локалізацію, масив сповіщень та дані локалізації. Сховище, описане в task-store.ts, призначено для задоволення потреб конкретної предметної області. Стани калькулятора винесені до окремого файлу calculator-store.ts. У кожному випадку поряд визначаються функції для більш зручного керування станами.

Використання селекторів станів має значні переваги перед вбудованим Context API:

— перемалювання компонентів відбувається лише в тому випадку, коли був оновлений пов’язаний із ними стан;

— немає потреби у визначенні обгорток для отримання доступу до даних.

У папці scss знаходяться описи стилів оформлення. Виокремлено піддиректорії, призначення кожної описано в таблиці (див. табл. 3.7).

Таблиця 3.7 — Піддиректорії папки scss

Директорія	**Призначення**
abstracts	Для абстрактних концепцій, що існують лише на рівні мови препроцесора: змінні та ін.
base	Базові спільні налаштування
components	Описи компонентів, які можна перевикористовувати
pages	Опис особливостей взаєморозташування компонентів на різних сторінках вебзастосунку
vendors	Підключення сторонніх бібліотек зі стилями

Структура фреймворку також включає папку utils, у якій розташовані кілька наборів допоміжних функцій. Призначення кожного описано в таблиці (див. табл. 3.8).

Таблиця 3.8 — Набори допоміжних функцій

Файл	**Призначення**
data.ts	Функції, що спрощують доступ до даних локалізації завдань: повертають функції-аксесори, побудовані за ідентифікатором завдання
generation.ts	Функції для роботи з випадковістю: генерація значень, перетасування масиву
http.ts	Функції для надіслання результатів на сервер
io.ts	Функція серіалізації об’єкта у файл JSON
validation.ts	Функції перевірки значень

Точкою входу в застосунок є index.tsx. У файлі react-app-env.d.ts наводяться оголошення додаткових типів ресурсів, що імпортуються (див. рис. 3.15). Це необхідно для їх включення в склад збірки.

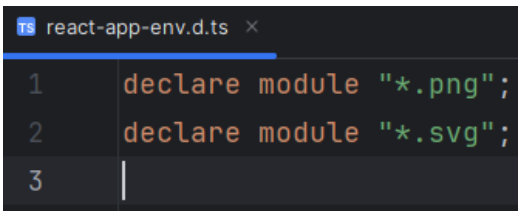


Рисунок 3.15 — Додаткові типи ресурсів для імпорту

3.2 Програмна реалізація основного тренажера

Конфігурація, configuration.ts, розширена: додані параметри, що накладають обмеження на ентропію генерації випадкової транспортної задачі. Можна налаштовувати мінімальну та максимальну кількість постачальників/споживачів, а також розміру тарифу на перевезення одиниці вантажу. Крім того підтримується налаштування кроку дискретизації, що враховується під час обрання числових значень.

Підтримує кілька режимів роботи: «Тренування», «Своя задача» та «Екзамен». У перших двох використовується толерантна до помилок модель оцінювання TrainingEvaluationModel, відображаються контекстні підказки та надається можливість перезапуску тренажера. Режим екзамену відрізняється використанням більш жорсткої моделі оцінювання KnowledgeEvaluationModel без підказок та можливості повернення на початок. Підтримується можливість введення статичної умови задачі, що не буде змінюватися.

Для покриття потреб предметної області додатково реалізовані моделі сутностей транспортної задачі (див. табл. 3.9), а також React-компоненти з відповідними дескрипторами для складових транспортної таблиці (див. табл. 3.10, 3.11). Хоч і з’являються паралельні ієрархії (компонент — дескриптор), утім перевагою такого підходу є зручність опису до елементів графічного інтерфейсу.

Таблиця 3.9 — Класи сутностей транспортної задачі

Клас	**Призначення**
Participant	Абстрактний клас учасника перевезень
Supplier	Постачальник

Закінчення таблиці 3.9

Клас	**Призначення**
Consumer	Споживач
Tariff	Тариф на перевезення

Таблиця 3.10 — Компоненти графічного інтерфейсу

Клас	**Призначення**
LabelCell	Клітина мітки учасника перевезень
PotentialCell	Клітина потенціалу учасника перевезень
SupplierLabelCell	Клітина мітки постачальника
SupplierProductCountCell	Клітина обсягу запасів постачальника
SupplierPotentialCell	Клітина потенціалу постачальника
ConsumerLabelCell	Клітина мітки споживача
ConsumerProductCountCell	Клітина обсягу потреб споживача
ConsumerPotentialCell	Клітина потенціалу споживача
TariffCell	Клітина тарифу на перевезення

Таблиця 3.11 — Класи дескрипторів для компонентів графічного інтерфейсу

Клас	**Призначення**
CellDescriptor	Абстрактний дескриптор клітини транспортної таблиці
SupplierCellDescriptor	Абстрактний дескриптор клітини постачальника
SupplierLabelCellDescriptor	Дескриптор клітини мітки постачальника
SupplierProductCountCellDescriptor	Дескриптор клітини обсягу запасів постачальника
SupplierPotentialCellDescriptor	Дескриптор клітини потенціалу постачальника

Закінчення таблиці 3.11

Клас	**Призначення**
ConsumerCellDescriptor	Абстрактний дескриптор клітини споживача
ConsumerLabelCellDescriptor	Дескриптор клітини мітки споживача
ConsumerProductCountCellDescriptor	Дескриптор клітини обсягу потреб споживача
ConsumerPotentialCellDescriptor	Дескриптор клітини потенціалу споживача
TariffCellDescriptor	Дескриптор тарифу на перевезення

Серед інших доданих компонентів: TaskTable — відповідає за компонування комірок транспортної таблиці, а також TaskTableToolBarButton і TaskTableToolBarFileButton — для керування даними в таблиці (збереження/завантаження/перегенерація).

Структура абстрактного класу Participant має вигляд (див. рис. 3.16).

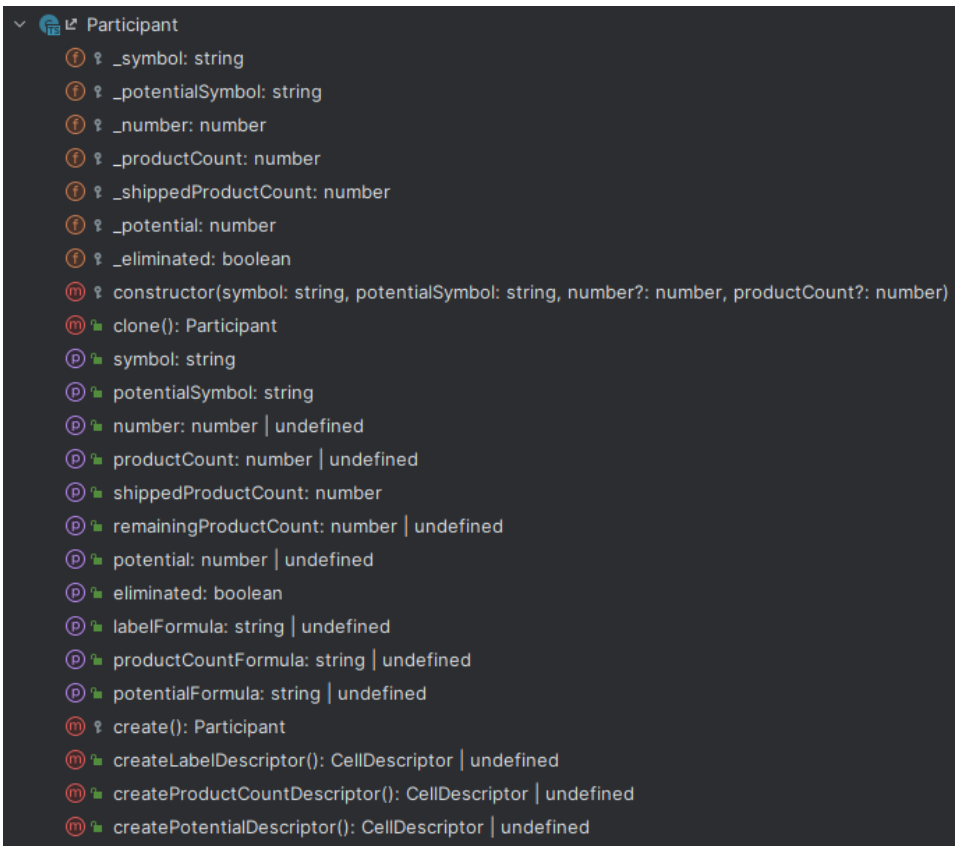


Рисунок 3.16 — Структура класу Participant

Символ мітки учасника _symbol (A — для постачальника, B — для споживача) та символ потенціалу _potentialSymbol (u — для постачальника та v — для споживача) встановлюється під час виклику конструктора. Дочірні класи, Supplier та Consumer, передають відповідні позначення, отримані з централізованої конфігурації, до конструктора батьківського класу.

Під час створення учасника його номер _number дозволяється не ініціалізувати, що корисно для ситуації, коли від користувача вимагається визначити номер фіктивного постачальника або споживача.

Під полем _productCount мається на увазі початковий обсяг запасів або потреб — інтерпретація залежить від типу учасника. Значення також можна не ініціалізувати, якщо проводиться приведення транспортної задачі до закритого типу.

Поле _shippedProductCount відображає обсяг вантажу, що вивезено/ввезено — інтерпретація залежить від типу учасника. Під час створення об’єкта ініціалізується значенням 0.

Оскільки в рамках тренажеру для розв’язання транспортної задачі застосовується виключно метод потенціалів, тому прийнято рішення зберігати значення потенціалу в об’єкті учасника перевезень у полі _potential. Воно може бути неініціалізованим, що дає змогу поставити запитання студенту.

Поле _eliminated визначає, чи усунуто учасника з розгляду в результаті вивезення всіх запасів / задоволення всіх потреб. За замовчуванням — false.

Крім того абстрактний клас Participant має кілька фабричних методів для зв’язування з дескрипторами: createLabelDescriptor, createProductCountDescriptor та createPotentialDescriptor. Крім того є приватний абстрактний метод create. Шляхом застосування патерну «шаблонний метод» реалізовано виведення LaTeX-формул для сутностей та клонування.

Структура класу тарифу (див. рис. 3.17).

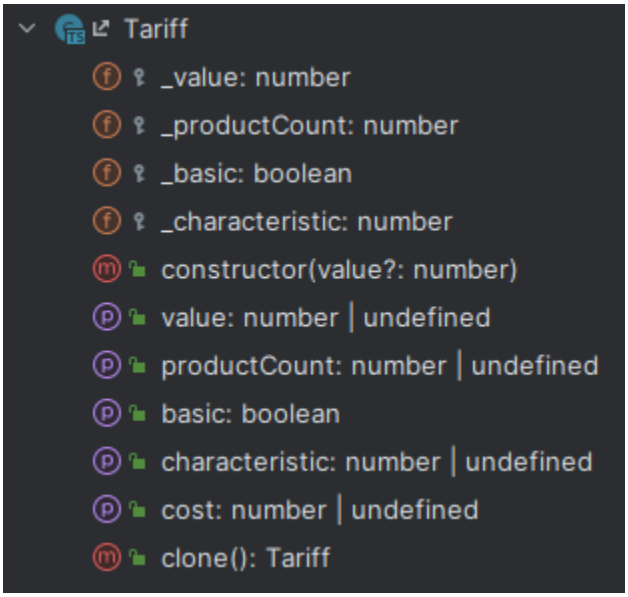


Рисунок 3.17 — Структура класу Tariff

Варто зазначити, що номер постачальника та номер споживача розглядається як мета-інформація, яку можна отримати виходячи з положення елементу в матриці. Значення тарифу _value може бути неініціалізованим, що необхідно для процедури приведення задачі до закритого типу, коли студенту треба визначитися, яка вартість перевезень одиниць вантажу між реальними учасниками перевезень та фіктивним.

Кількість перевезеного вантажу _productCount також змінюється динамічно. На початку дорівнює undefined.

Інформація про базисність (_basic) зберігається в об’єкті цієї клітини. Для небазисних клітин обчислюється характеристика вільної клітини _characteristic.

Був доданий TransitionCycleTask — новий тип завдання, суть якого полягає в побудові циклу переходу. До того ж створений клас відповіді, TransitionCycleAnswer, що зберігає всередині себе координати вузлів ламаної. Цими координатами є відповідні за комірками тарифів, через які проходить ламана, номери постачальників та споживачів.

Базових статичних ресурсів повністю вистачило, окрім кривої — роздільника для кутової клітини. Вона створювалася окремо за допомогою Adobe Illustrator.

Flowable-елементи розташовані один в одному (див. додаток З) у відповідності до попередньо спроєктованого n-арного дерева з керувальними вузлами (див. рис. 2.2). Завдання тренажера являють собою 34 окремих TypeScript-файли, що експортують константи до загального об’єкта task-flow, що є лінійною послідовністю.

Перелік допоміжних функцій був доповнений такими, що відповідають за генерацію транспортної задачі відповідно до конфігурації тренажера. Також є ті, що дозволяють зручно ітеруватися не тільки масивами постачальників/споживачів (forEachSupplier, forEachConsumer), а й матрицею тарифів (forEachTariff).

Серед сховищ змін зазнало тільки TaskStore: додані поля для роботи з даними транспортної задачі (див. табл. 3.12).

Таблиця 3.12 — Поля сховища для роботи з даними транспортної задачі

Поле сховища	**Тип**	**Призначення**
suppliers	Supplier[]	Поточний масив постачальників
consumers	Consumer[]	Поточний масив споживачів
tariffs	Tariff[][]	Поточна матриця тарифів
setSuppliers	(suppliers: Supplier[]) => void	Оновлення масиву постачальників шляхом повної заміни
setConsumers	(consumers: Consumer[]) => void	Оновлення масиву споживачів шляхом повної заміни
setTariffs	(tariffs: Tariff[][]) => void	Оновлення матриця тарифів шляхом повної заміни
updateSupplier	(supplierIndex: number, updatedSupplier: Supplier) => void	Оновлення постачальника
updateConsumer	(consumerIndex: number, updatedConsumer: Consumer) => void	Оновлення споживача

Закінчення таблиці 3.12

Поле сховища	**Тип**	**Призначення**
updateParticipant	(participantIndex: number, updatedParticipant: Participant) => void	Оновлення учасника перевезень
deleteSupplier	(supplierIndex: number) => void	Видалення постачальника
deleteConsumer	(consumerIndex: number) => void	Видалення споживача
updateTariff	(supplierIndex: number, consumerIndex: number, updatedTariff: Tariff) => void	Оновлення тарифу
updateTariffs	(participantType: ParticipantType, participantIndex: number, updatedTariffs: Tariff[]) => void	Оновлення частини тарифів (за постачальником або споживачем)
Selection	CellDescriptor[]	Поточне відображуване виділення користувача
setSelection	(...cellDescriptors: CellDescriptor[]) => void	Перевизначення виділення
select	(...cellDescriptors: CellDescriptor[]) => void	Додавання до виділення
clearSelection	() => void	Очищення виділення
cycle	TransitionCycleAnswer	Поточний відображуваний цикл переходу
setCycle	(transitionCycleAnswer: TransitionCycleAnswer) => void	Перевизначення циклу переходу

JSON файли локалізацій та схеми доповненні необхідними перекладами відповідно до предметної області задачі, процес розв’язання якої моделювався.

3.3 Програмна реалізація альтернативного тренажера

Альтернативний тренажер створений із використанням фреймворку jTrainer. Обрана технологія функціонує на базі звичайної автоматної моделі, де попередня історія взаємодії користувача з навчальним об’єктом не враховується, не підтримуються розгалуження та цикли — доступна тільки лінійна послідовність завдань.

Структура проєкту має вигляд (див. рис. 3.18).

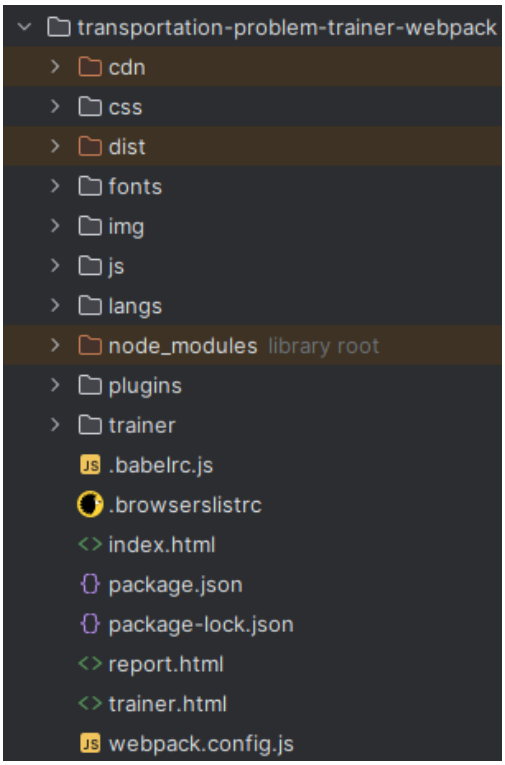


Рисунок 3.18 — Структура проєкту альтернативного тренажера

Це дещо модифікований шаблон: введено використання збиральника Webpack та транспілятора Babel, щоб мати змогу писати JavaScript із використанням сучасного синтаксису. Модель функціонування не змінена.

У файлі package.json є інформація про даний пакет: назва, версія тощо. Основні runtime-залежності відсутні, оскільки даний тренажер спирається на фреймворк jTrainer, що отримуватиметься від CDN.

Залежності розробки (devDependencies) проєкту описані в таблиці (див. табл. 3.13).

Таблиця 3.13 — Залежності проєкту

Залежність	**Призначення**
@babel/core	Ядро Babel: містить основні функції транспіляції
@babel/preset-env	Плагін для Babel, який дозволяє автоматично визначати необхідні трансформації коду для підтримки потрібних версій браузерів
Archiver	Пакет для створення архівів ZIP
babel-loader	Завантажувач Babel для Webpack
browserslist	Інструмент для опису переліку підтримуваних браузерів
clean-webpack-plugin	Плагін для Webpack, шо дозволяє очищати цільову директорію перед формуванням збірки
copy-webpack-plugin	Плагін для Webpack, що дозволяє копіювати файли без змін
core-js	Сучасна бібліотека поліфілів
webpack	Збиральник модулів
webpack-cli	Інтерфейс командного рядка для Webpack

У файлі .babelrc.js наведені налаштування транспіляції із використанням Babel (див. рис. 3.19).

Рисунок 3.19 — Налаштування транспіляції

Файл .browserslistrc призначений для опису підтримуваних версій браузерів (див. рис. 3.20). Уміст сформований за допомогою сервісу browsersl.ist.

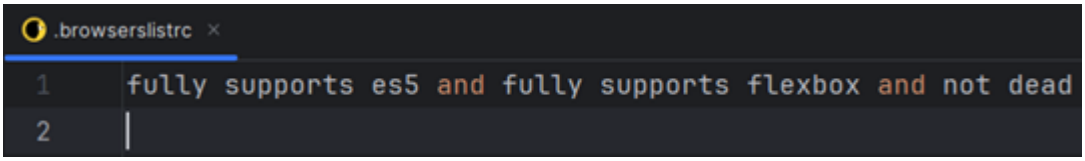


Рисунок 3.20 — Опис переліку підтримуваних браузерів

Налаштування Webpack є в файлі webpack.config.js (див. рис. 3.21).

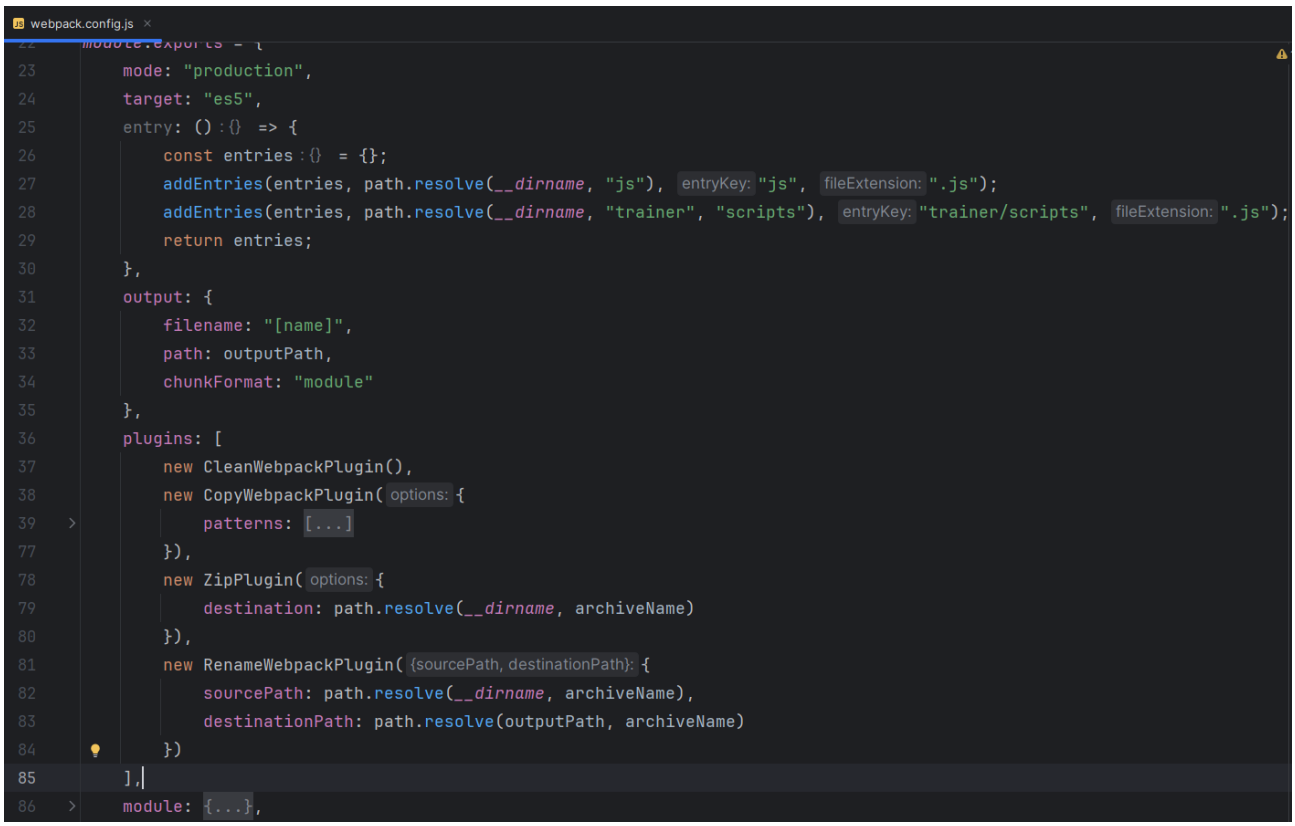


Рисунок 3.21 — Конфігурація Webpack

Особливістю збирання є велика кількість точок входу: кожен сценарій JavaScript розглядається окремо. У такий спосіб досягаємо приведення коду до старого стандарту без порушення логіки функціонування jTrainer. Цей фреймворк розроблявся у такий спосіб, щоб завантаження всіх необхідних кроків відбувалося динамічно, тому Webpack застосовується як зручний інструмент опису перетворень.

Додатково розроблені плагіни для Webpack: ZipPlugin — для пакування вмісту цільової директорії до архіву ZIP (див. рис. 3.22) та RenameWebpackPlugin — щоб виконувати перейменування файлів для переміщення (див. рис. 3.23). Вони розташовані в директорії plugins.

Файл index.html містить iframe-обгортку для trainer.html — основної сторінки тренажера, уміст аналогічний (див. рис. 3.4). Файл report.html — порожній: він є заглушкою, що дозволяє пройти перевірку під час завантаження навчального застосунку на сервер.

Рисунок 3.22 — Webpack-плагін для пакування до архіву ZIP

Рисунок 3.23 — Webpack-плагін для перейменування файлів

Піддиректорії проєкту призначені для збереження статичних ресурсів (див. табл. 3.14).

Таблиця 3.14 — Призначення піддиректорій проєкту

Директорія	**Уміст**
css	Додаткові стилі
fonts	Шрифти
img	Зображення
js	Додаткові сценарії JavaScript
langs	JSON файли локалізацій
plugins	Плагіни для Webpack
trainer	Основний контент тренажера

До основного контенту належать html-шаблони кроків (обробляються шаблонізатором Mustache), їхня логіка в піддиректорії scripts (сценарії JavaScript) та налаштування тренажера в папці settings.

Тренажер складається з 28 кроків, серед яких 26 є завданнями. Під час розроблення цього тренажера метою було забезпечити повноту розгляду теми «Транспортні задачі». Ключовою відмінністю альтернативного тренажера від основного був процес розв’язання задачі, що пояснюється відмінностями використовуваних автоматних моделей. Розв’язання транспортної задачі за основним тренажером є неперервним, коли як альтернативний пропонує фрагментарне ознайомлення на основі непов’язаних між собою ситуацій.

Отже, цей тренажер являє собою тест із завданнями, що генеруються динамічно (див. додаток И). Для їх формування знадобилося реалізувати алгоритм генерації транспортних задач, де враховуються межі ентропії, задані параметрами (див. рис. 3.24):

— мінімальна кількість постачальників (minSupplierCount);
— максимальна кількість постачальників (maxSupplierCount);
— мінімальний обсяг запасів одного постачальника (minSupplierProductCount);
— максимальний обсяг запасів одного постачальника (maxSupplierProductCount);
— крок обсягу запасів постачальника (supplierProductCountStep);
— мінімальна кількість споживачів (minConsumerCount);
— максимальна кількість споживачів (maxConsumerCount);
— мінімальний обсяг потреб одного споживача (minConsumerProductCount);
— максимальний обсяг потреб одного споживача (maxConsumerProductCount);
— крок обсягу потреб споживача (consumerProductCountStep);
— мінімальне значення тарифу (minTariffValue);
— максимальне значення тарифу (maxTariffValue);
— крок значення тарифу (tariffValueStep).

```
static generate(
    minSupplierCount, maxSupplierCount, minSupplierProductCount, maxSupplierProductCount, supplierProductCountStep,
    minConsumerCount, maxConsumerCount, minConsumerProductCount, maxConsumerProductCount, consumerProductCountStep,
    minTariffValue, maxTariffValue, tariffValueStep, open = undefined
) {
    let transportationProblem;
    do {
        const suppliers = jTrainerExt.generateSuppliers(minSupplierCount, maxSupplierCount, minSupplierProductCount, maxSu
        const consumers = jTrainerExt.generateConsumers(minConsumerCount, maxConsumerCount, minConsumerProductCount, maxCo
        const tariffs = jTrainerExt.generateTariffs(suppliers.length, consumers.length, minTariffValue, maxTariffValue, ta
        transportationProblem = new jTrainerExt.TransportationProblem(suppliers, consumers, tariffs);
    } while (open !== undefined && transportationProblem.open !== open);
    return transportationProblem;
}
```

Рисунок 3.24 — Метод для генерації транспортної задачі

Додатково можна вимагати певний тип задачі (параметр open). За замовчуванням значення дорівнює undefined, тобто він не є важливим. Якщо ж встановити true або false, тоді генерація проводитиметься доти, поки не буде знайдена така умова, що відповідає вимогам.

Розглянутий алгоритм підходить для швидкого створення простих задач, утім цього недостатньо для повноцінного розкриття теми «Транспортні задачі». До прикладу, якщо треба знайти ситуацію, коли побудована ламана циклу переходу містить кілька парних вузлів (зі знаком «−»), за якими обсяги поставок однакові. Щоб знайти необхідну умову задачі, треба провести велику кількість обчислень, що є контрпродуктивним, оскільки це лише одна задача тесту.

Більш ефективний підхід — проводити генерацію транспортної задачі та її розв’язання в очікуванні виконання умови пошуку. Якщо спроба не була успішною — перейти до наступної ітерації, поки потрібна транспортна задача не буде знайдена. За використання такого підходу важливо подбати про достатній рівень ентропії, щоб отримання необхідної транспортної задачі було цілком ймовірним.

Важливо подбати про механізм зупинки процесу розв’язання. Це важливо в двох випадках:

— якщо можна відкинути результати генерації достроково, щоб швидше перейти до наступної;

— коли треба надати користувачу конкретну ситуацію з процесу розв’язання.

Для цього прийнято рішення використовувати «точки зупину» — моменти виклику функції-предикату в рамках функції розв’язання транспортних задач. Перевірки розставляються перед ділянками коду, що призводять до зміни стану задачі.

Крім того важливо враховувати детермінованість процесу розв’язання. До прикладу, як тільки трапляється варіативність: усунути з розгляду споживача або постачальника, видалити з базису одну клітину або іншу в результаті застосування циклу переходу тощо.

Перелік завдань, що генеруються, наведений у таблиці (див. табл. 3.15).

Таблиця 3.15 — Опис завдань тренажера

Крок	**Вимоги до генерації**	**Завдання**
Вступ	—	—
1	Транспортна задача довільного типу	Ввести кількість постачальників, споживачів, загальний обсяг запасів та потреб
2	5 транспортних задач (щонайменше по одній кожного типу)	Визначити тип кожної транспортної задачі
3	—	Побудувати правильні твердження стосовно приведення транспортної задачі до закритого типу
4	Дві транспортні задачі: перша з фіктивним споживачем, а друга — з фіктивним постачальником	Знайти потреби/запаси фіктивних учасників перевезень
5	—	Обчислення необхідної кількості базисних клітин за теоремою про ранг
6	Транспортна задача без фіктивних учасників, для якої можна однозначно побудувати початковий опорний план за методом північно-західного кута	Установлення послідовності розгляду клітин тарифів транспортної таблиці за методом північно-західного кута

Продовження таблиці 3.15

Крок	**Вимоги до генерації**	**Завдання**
7	Транспортна задача без фіктивних учасників, для якої можна однозначно побудувати початковий опорний план за методом мінімальної вартості	Установлення послідовності розгляду клітин тарифів транспортної таблиці за методом мінімальної вартості
8	Транспортна задача без фіктивних учасників, знайдена перша базисна комірка	Ввести обсяг поставки, оновлені запаси постачальника та потреби споживача
9	Транспортна задача, де для поточного постачальника раніше була знайдена інша базисна клітина	Ввести обсяг поставки, оновлені запаси постачальника та потреби споживача
10	Транспортна задача, де для поточного споживача раніше була знайдена інша базисна клітина	Ввести обсяг поставки, оновлені запаси постачальника та потреби споживача
11	Транспортна задача, де після виконання поставки треба видалити або постачальника, або споживача	Обрати варіант, який відображає всі можливі дії
12	Транспортна задача, де після виконання поставки треба видалити постачальника та споживача	Обрати варіант, який відображає всі можливі дії
13	Транспортна задача, де після виконання поставки треба видалити постачальника	Обрати варіант, який відображає всі можливі дії
14	Транспортна задача, де після виконання поставки треба видалити споживача	Обрати варіант, який відображає всі можливі дії
15	Транспортна задача з побудованим опорним планом	Обчислити загальну вартість перевезень
16	—	Обрати правильні твердження щодо застосування методу потенціалів для розв’язання транспортної задачі

Продовження таблиці 3.15

Крок	**Вимоги до генерації**	**Завдання**
17	Транспортна задача на початку застосування методу потенціалів, $u_1 = 0$, наступні потенціали (споживача та постачальника) можна знайти послідовно	Обчислити значення потенціалів споживача та постачальника
18	Транспортна задача з побудованим початковим опорним планом	Обрати вільні клітини
19	Транспортна задача з визначеними для учасників потенціалами	Обчислити значення характеристики вільної клітини
20	Транспортна задача з визначеними характеристиками вільних клітин (усі додатні)	Обрати правильні твердження стосовно наведеного плану перевезень
21	Транспортна задача з визначеними характеристиками вільних клітин (усі невід'ємні: є нуль)	Обрати правильні твердження стосовно наведеного плану перевезень
22	Транспортна задача з визначеними характеристиками вільних клітин (є від'ємні)	Обрати правильні твердження стосовно наведеного плану перевезень
23	6 транспортних задач із побудованим циклом переходу: — будь-яка транспортна задача з циклом переходу (правильний); — транспортна задача з циклом переходу з самоперетином (правильний); — транспортна задача з циклом переходу з неправильною послідовністю знаків «+» та «−»; — транспортна задача з циклом переходу, що починається не з комірки з найменшою характеристикою; 2 транспортні задачі з циклом переходу, що побудований не за базисними клітинами	Обрати правильно побудовані цикли переходу

Закінчення таблиці 3.15

Крок	**Вимоги до генерації**	**Завдання**
24	Транспортна задача з побудованим циклом переходу	Обчислити обсяг перерозподілу для побудованого циклу переходу
25	Транспортна задача з побудованим циклом переходу з 4 вузлами та однією клітиною, що має покинути базис	Надати правильні твердження щодо результатів застосування побудованого циклу переходу
26	Транспортна задача з побудованим циклом переходу з 4 вузлами та двома клітинами, одна з яких має покинути базис	Надати правильні твердження щодо результатів застосування побудованого циклу переходу
Результати	—	—

Описи завдань наводяться в узагальненому вигляді, без конкретних варіантів відповідей.

3.4 Тестування навчальних об’єктів

Для перевірки правильності виконання тренажерів на визначених у якості мінімальних версіях браузерів проведено ручне тестування.

Приклад роботи основного тренажера в браузері Firefox 78 (див. рис. 3.25) та альтернативного — у браузері Chrome 23 (див. рис. 3.26).

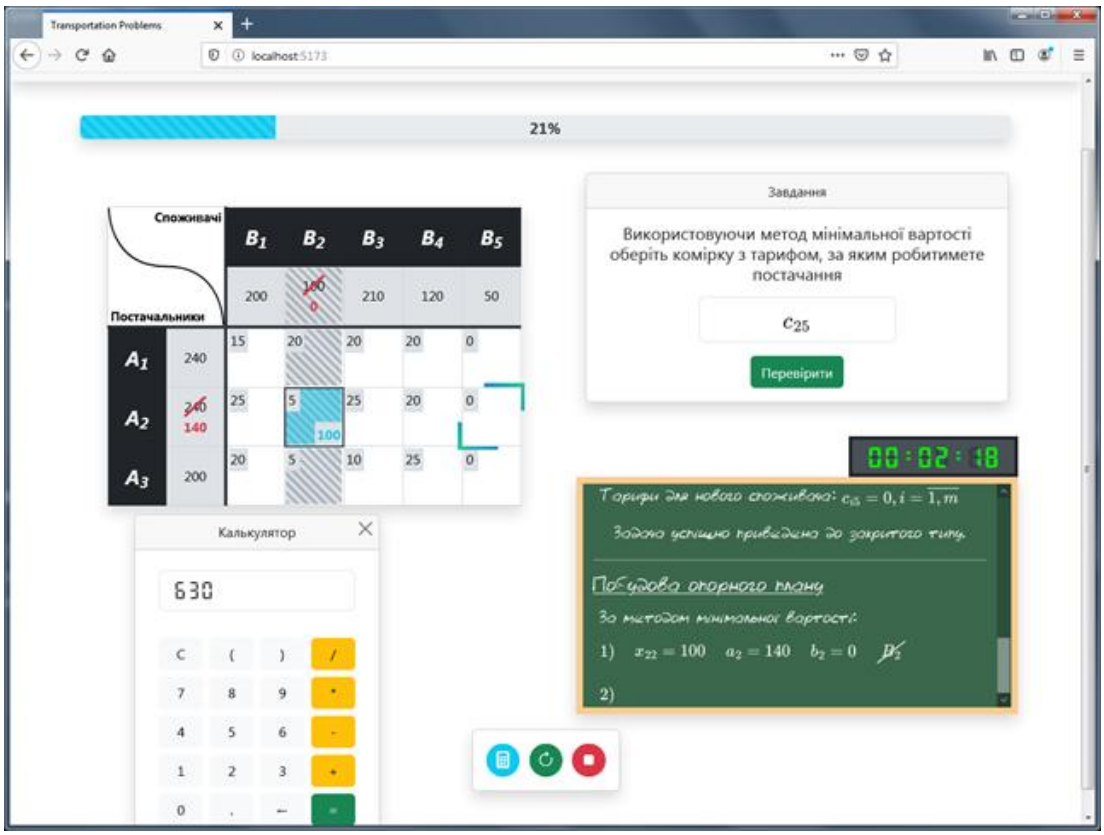


Рисунок 3.25 — Тестовий запуск основного тренажера в Firefox 78

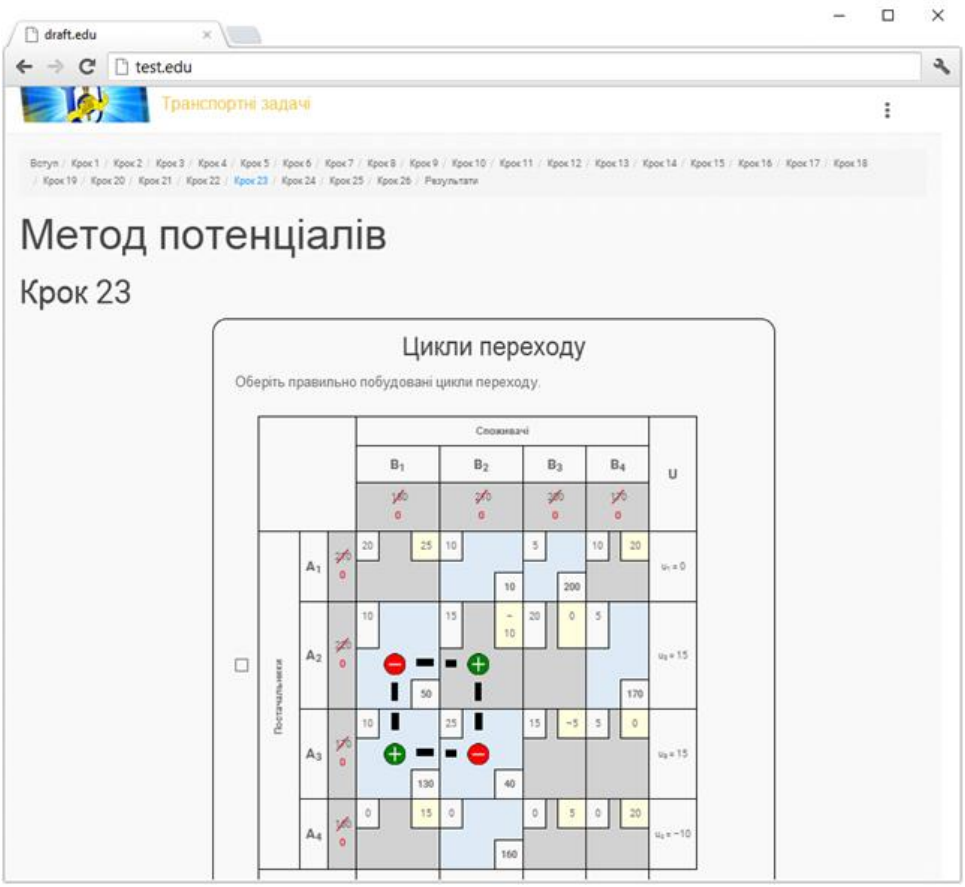


Рисунок 3.26 — Тестовий запуск альтернативного тренажера в Chrome 23

Ручне тестування проведено успішно: основний тренажер працює коректно в браузерах, наведених у таблиці 2.5. Помилок виконання JavaScript-сценаріїв не виявлено. Зовнішній вигляд інтерфейсу користувача відповідає очікуванням. Альтернативний тренажер — аналогічно в браузерах (див. табл. 2.6).

3.5 Аналіз результатів освітнього експерименту

Структура Python-проєкту для статистичного аналізу даних наведена на рисунку 3.27. У resources розташовані таблиці з вхідною інформацією про учасників експерименту: результати анкетування завантажені з Google Forms, а дані про активність на платформі Mix (participants.csv) зібрані з використанням спеціально розробленого інструмента для браузера (див. додаток К.1). Сценарій transform_csv.py (див. додаток К.2) виконує консолідацію даних до data.csv.

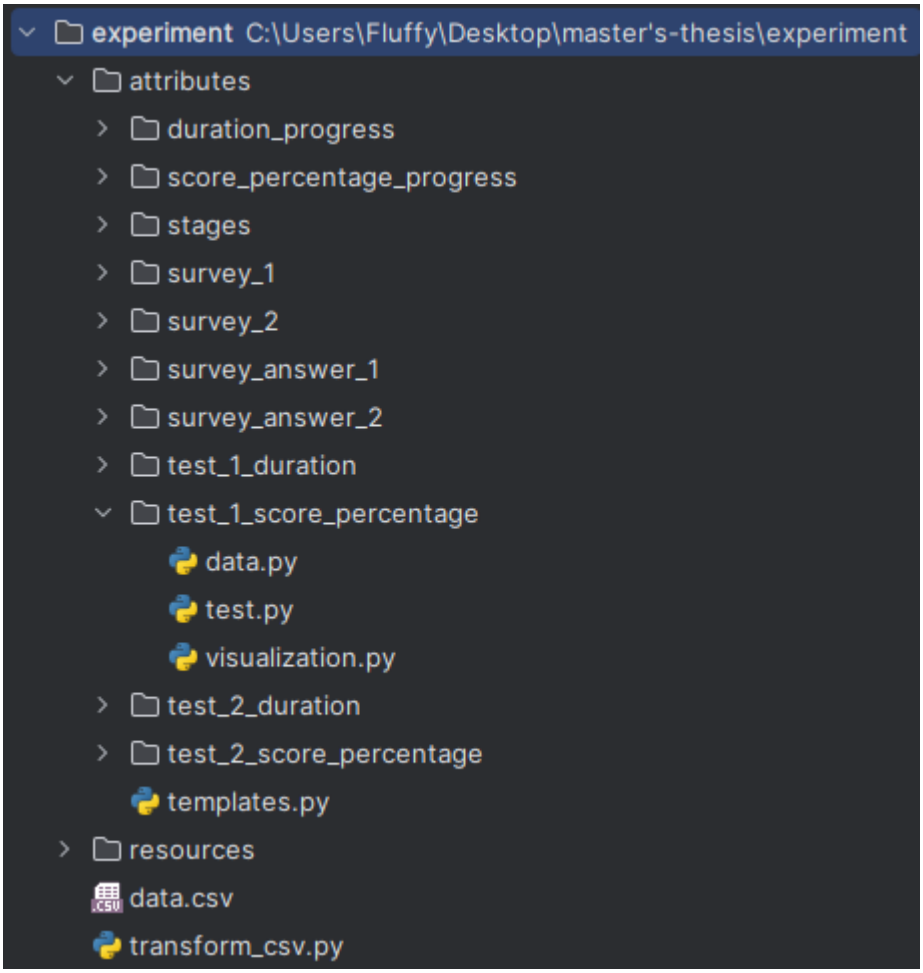


Рисунок 3.27 — Структура Python-проєкту для аналізу результатів

Аналіз проводився в кілька кроків. Для кожного створена директорія в attributes за структурою: data.py — опис доступу до даних, test.py — перевірка статистичних гіпотез (за необхідності) та visualization.py — для побудови діаграми. Спільний код винесений до templates.py (див. додаток К.3).

Кількість учасників за етапами наведена на рисунку 3.28. Очікувалося, що відвідуваність занять дисципліни «Математичні методи дослідження операцій» є рівномірною, утім спостерігалася конверсія. Найменша — на етапі поточного тестування: 68% для експериментальної та 32% для контрольної. Серед ймовірних причин: участь на вільних засадах та нестабільна ситуація в країні.

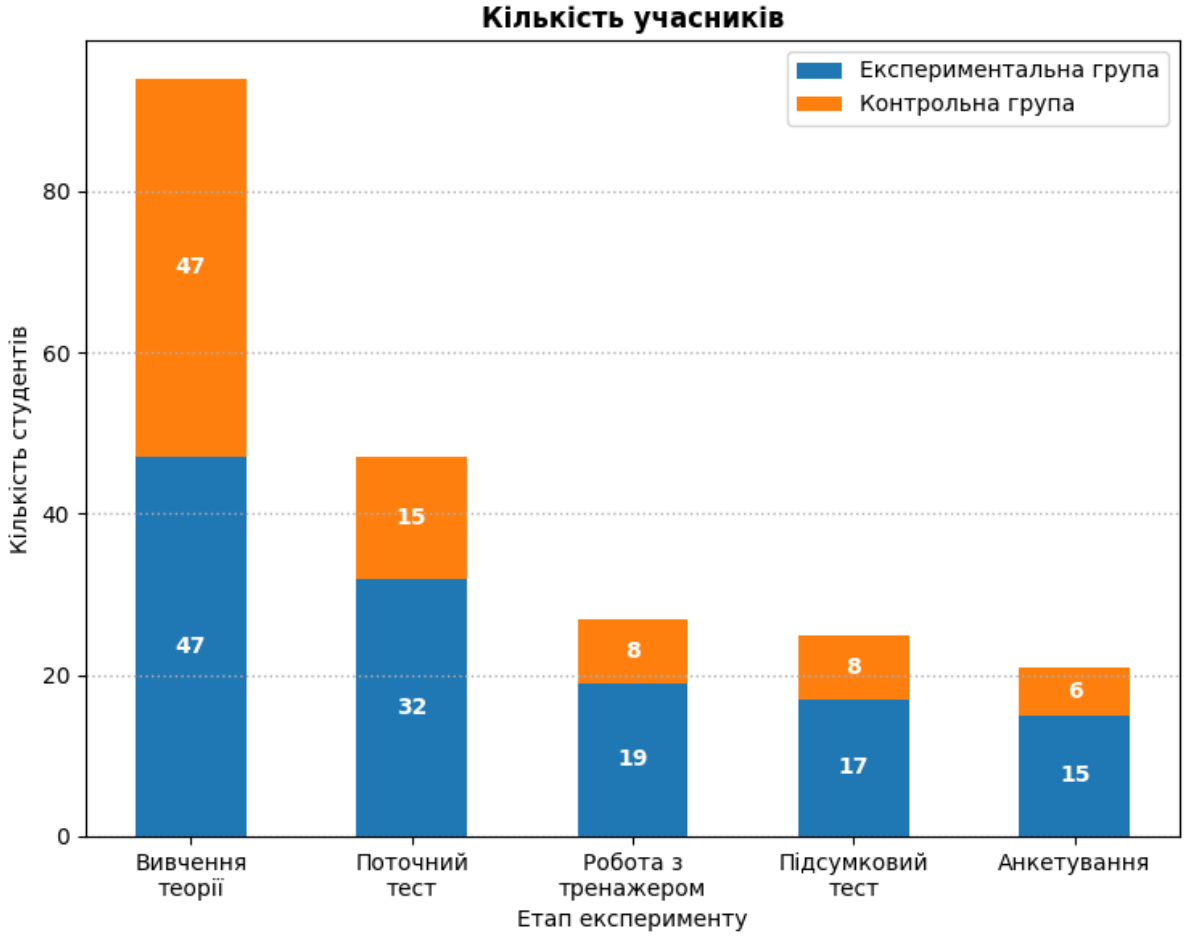


Рисунок 3.28 — Кількість учасників за етапами експерименту

Розподіли оцінок за поточний тест для обох груп наведені на рисунку 3.29. Закон розподілу оцінок експериментальної групи відмінний від нормального: прийнята альтернативна гіпотеза H_1, оскільки за критерієм Ройстона Шапіро-

Уілка $p = 0.003$, що менше за рівень значущості $\alpha = 0.05$. Оцінки контрольної групи розподілені нормально: прийнята гіпотеза H_0, оскільки обчислене за тим же критерієм значення $p = 0.117$ є більшим за рівень значущості α. Статистично значуща різниця між вибірками відсутня: прийнята гіпотеза H_0, оскільки за U-критерієм Манна-Уітні отримали $p = 0.418$, що більше за рівень значущості α.

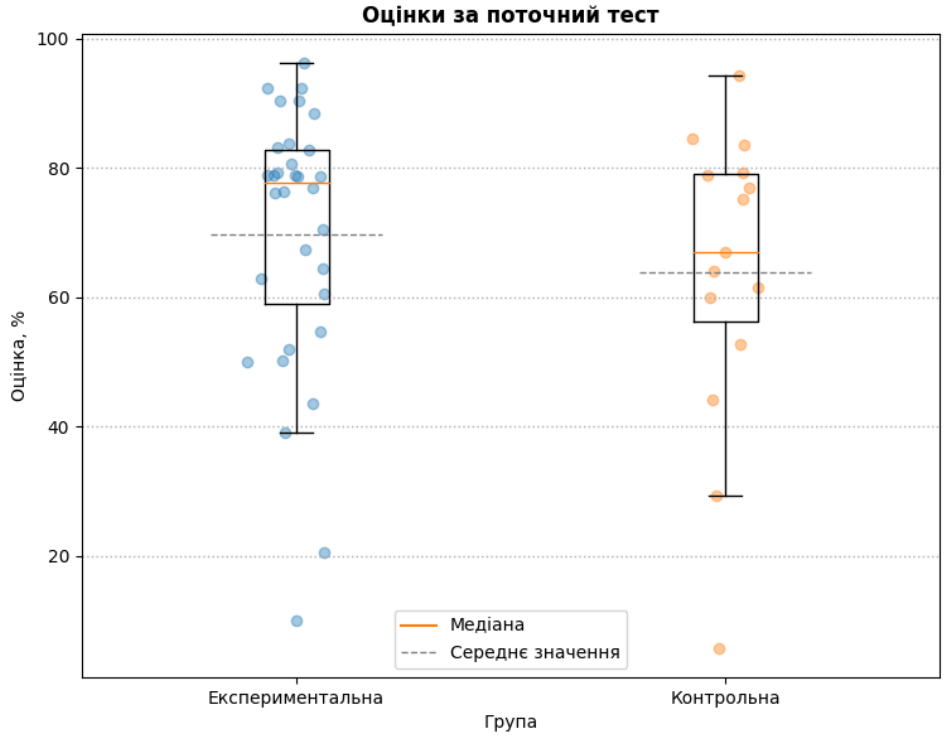


Рисунок 3.29 — Оцінки за поточний тест у групах

Часові витрати на виконання поточного тесту за групами учасників позначені на рисунку 3.29. Вони підкорюються нормальному закону розподілу: для експериментальної та контрольної групи прийнята гіпотеза H_0, оскільки $p = 0.069$ та $p = 0.695$ (за критерієм Ройстона Шапіро-Уілка) відповідно, що більше за α. Вибіркова дисперсія експериментальної групи $s_1^2 = 289.448$, а контрольної — $s_2^2 = 341.552$. Прийнята гіпотеза H_0 про рівність дисперсій у результаті застосування двостороннього тесту Фішера: розрахункове значення критерію $F = 1.180$ не належить критичним областям $(-\infty; 0.367)$ та $(2.321; +\infty)$. Середні значення часових витрат $\bar{x}_1 = 38.312$ й $\bar{x}_2 = 41.533$ експериментальної та

контрольної групи відповідно. У результаті застосування t-критерію Стьюдента для двох незалежних вибірок із рівними дисперсіями (2.25) прийнята гіпотеза H_0 про рівність середніх: $p = 0.559$, що більше за рівень значущості α.

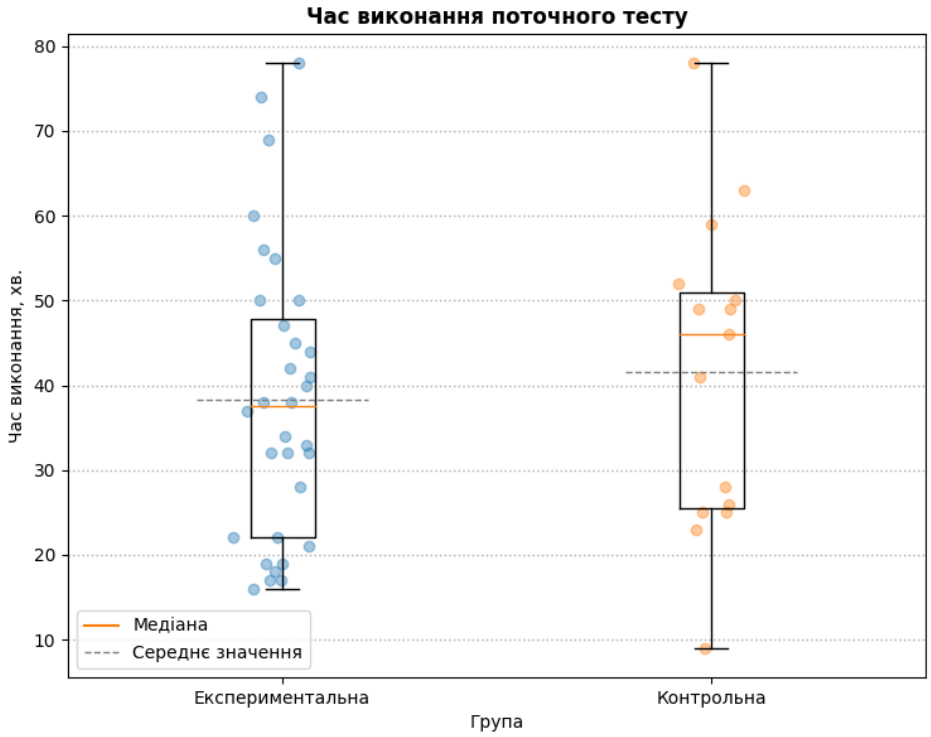


Рисунок 3.30 — Час виконання поточного тесту за групами

Незважаючи на відмінність у розмірах груп, вони лишилися репрезентативними: оцінки за поточне тестування (див. рис. 3.29) та відповідні часові витрати (див. рис. 3.30) були співставними, тобто не містили статистично значущих різниць.

Використання тренажерів в експериментальній та контрольній групі мало позитивний вплив на учасників.

Абсолютні прирости оцінок обох груп (див. рис. 3.31) розподілені нормально: прийнята гіпотеза H_0 про нормальний розподіл, оскільки $p > \alpha$ виконується для цих вибірок ($p = 0.388$ для експериментальної та $p = 0.577$ — для контрольної). Дисперсія приростів оцінок для експериментальної групи $s_1^2 = 95.824$, а контрольної — $s_2^2 = 122.330$. Прийнята гіпотеза H_0 про рівність

дисперсій за використання двостороннього тесту Фішера: розрахункове значення $F = 1.277$ не належить критичним областям $(-\infty; 0.220)$ та $(3.219; +\infty)$. Середні значення для експериментальної та контрольної групи: $\bar{x}_1 = 9.294\,\%$ та $\bar{x}_2 = 16.587\,\%$ відповідно. У результаті застосування двостороннього t-критерію Стьюдента для двох незалежних вибірок прийнято гіпотезу H_0 про $\bar{x}_1 = \bar{x}_2$, оскільки $p = 0.109$ більше за рівень значущості α.

Примітка. Відсотки вживаються в значенні розмірності для кількості балів.

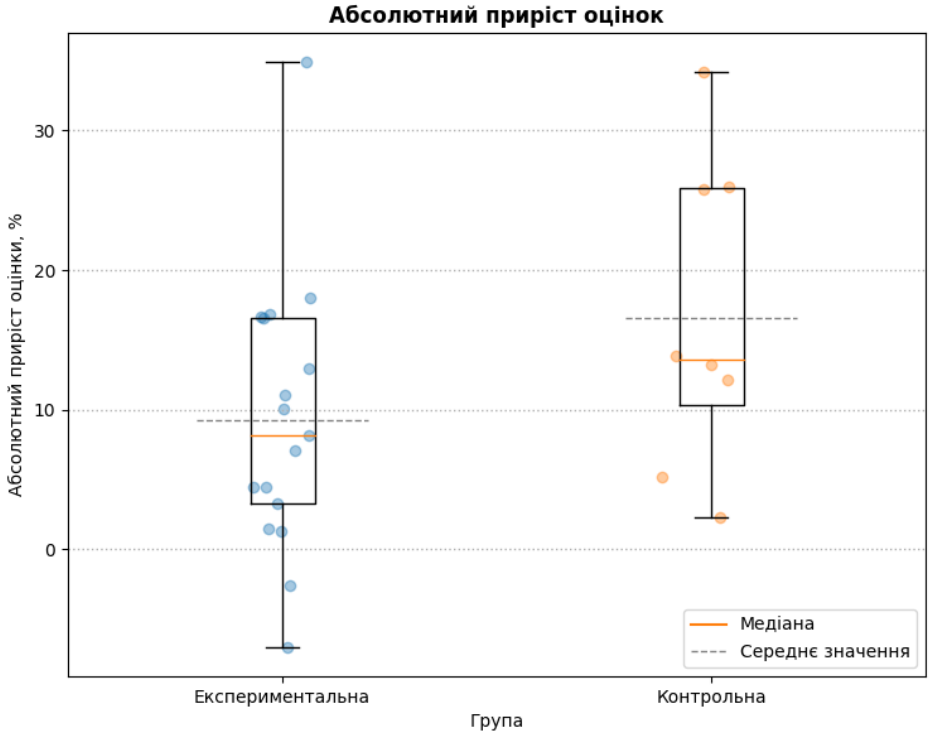


Рисунок 3.31 — Абсолютний приріст оцінок у групах

Економія часу учасниками обох груп відображена на рисунку 3.32. Значення для експериментальної розташовані за законом, що відмінний від нормального: прийнята альтернативна гіпотеза H_1, оскільки за використання критерію Ройстона Шапіро-Уілка $p = 0.023$, що менше за α. Економія часу для контрольної групи розподілена нормально: прийнята гіпотеза H_0, оскільки обчислене за тим же критерієм $p = 0.299$. За U-критерієм Манна-Уітні $p = 0.641$ — прийнята

гіпотеза H_0 про відсутність статистично значущої різниці між економією часу двома групами.

Середній зекономлений час на виконання тесту для експериментальної групи склав $\overline{x}_1 = 14.529$ хв., а для контрольної — $\overline{x}_2 = 17.125$ хв.

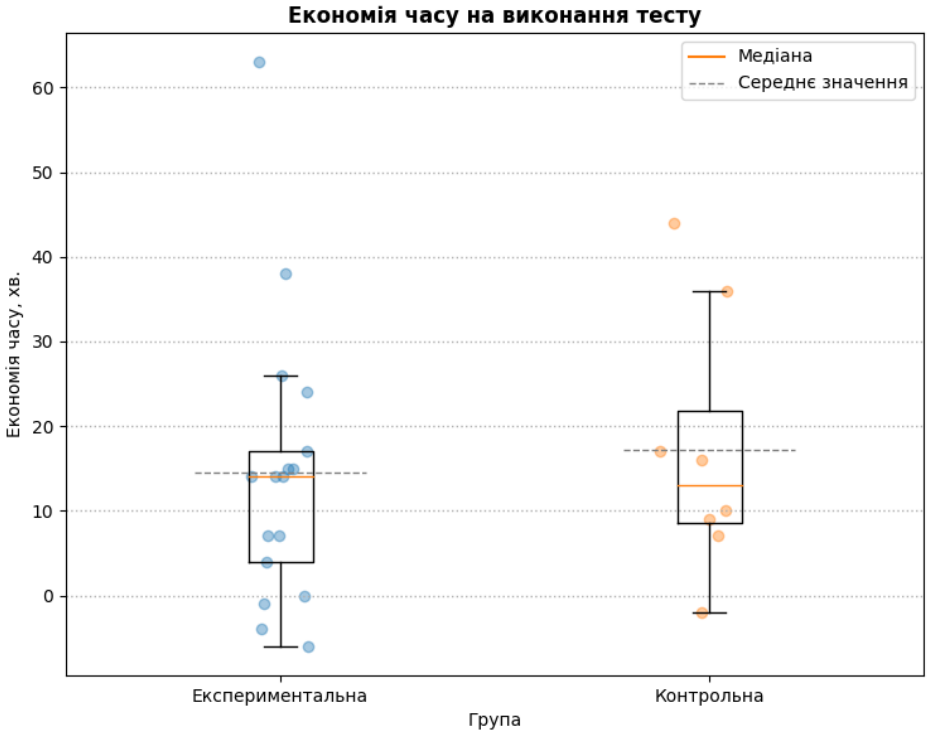


Рисунок 3.32 — Економія часу на виконання тесту за групами

Додатково проаналізовані абсолютні значення оцінок (див. рис. 3.33) та часових витрат наприкінці експерименту (див. рис. 3.34).

Оцінки за підсумковий тест кожної групи (див. рис. 3.33) підкорюються нормальному закону розподілу: прийнята гіпотеза H_0 для експериментальної та контрольної групи, оскільки в результаті застосування критерію Ройстона Шапіро-Уілка отримали $p = 0.334$ та $p = 0.155$ відповідно, що більше за α. Дисперсії оцінок у кожній із груп: $s_1^2 = 30.126$ — для експериментальної та $s_2^2 = 112.203$ — для контрольної. На основі двостороннього тесту Фішера прийнята гіпотеза H_1 про $s_1^2 \neq s_2^2$, оскільки розрахункове $F = 3.724$ належить правій критичній області $(3.219; +\infty)$. Середня оцінка для експериментальної групи $\overline{x}_1 = 89.247$, а для

контрольної — $\bar{x}_2 = 86.262$. За двостороннім t-критерієм Стьюдента прийнята гіпотеза H_1 про відсутність статистично значущої різниці, оскільки $p = 0.472$, що більше за рівень значущості α.

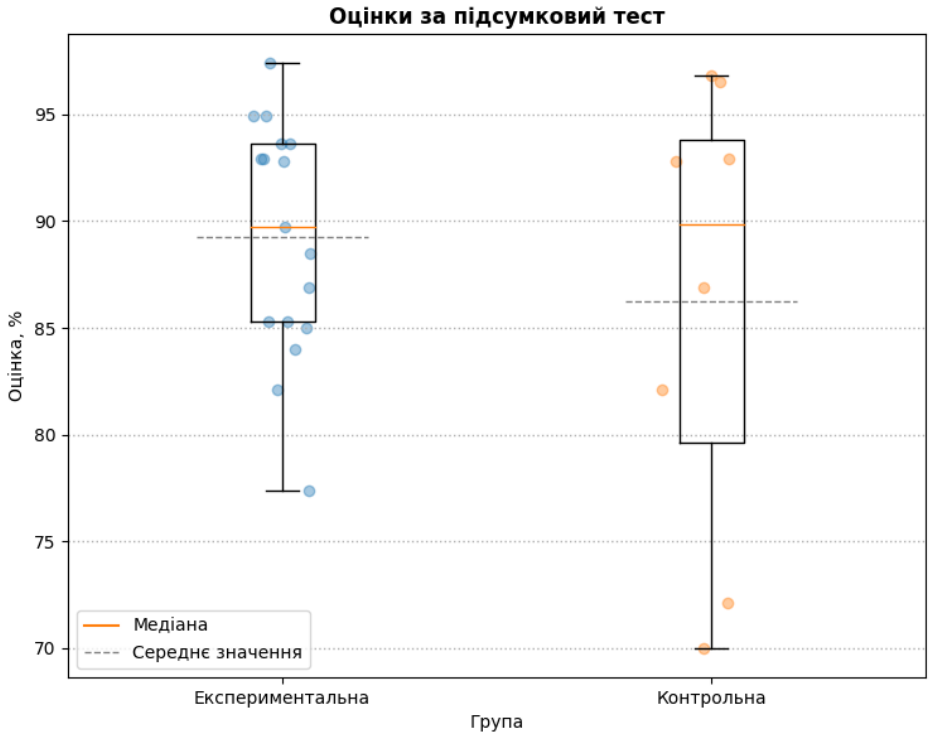


Рисунок 3.33 — Оцінки за підсумковий тест за групами

Часові витрати під час виконання підсумкового тесту (див. рис. 3.34) розподілені нормально: прийнята гіпотеза H_0, бо за критерієм Ройстона Шапіро-Уілка $p = 0.340$ для експериментальної групи та $p = 0.795$ — для контрольної. Відповідні дисперсії часових витрат: $s_1^2 = 55.066$ та $s_2^2 = 75.714$. Прийнята гіпотеза H_0 про $s_1^2 = s_2^2$ на основі двостороннього тесту Фішера: розрахункове значення критерію $F = 1.375$ не потрапило до критичної області $(-\infty; 0.220)$ та $(3.219; +\infty)$. Середні значення: $\bar{x}_1 = 22.235$ хв. для експериментальної та $\bar{x}_2 = 28$ хв. — для контрольної. Прийнята гіпотеза H_1 про $\bar{x}_1 < \bar{x}_2$, оскільки за використання лівостороннього t-критерію Стьюдента для незалежних вибірок $p = 0.050$, що дорівнює рівню значущості α (див. табл. 2.7).

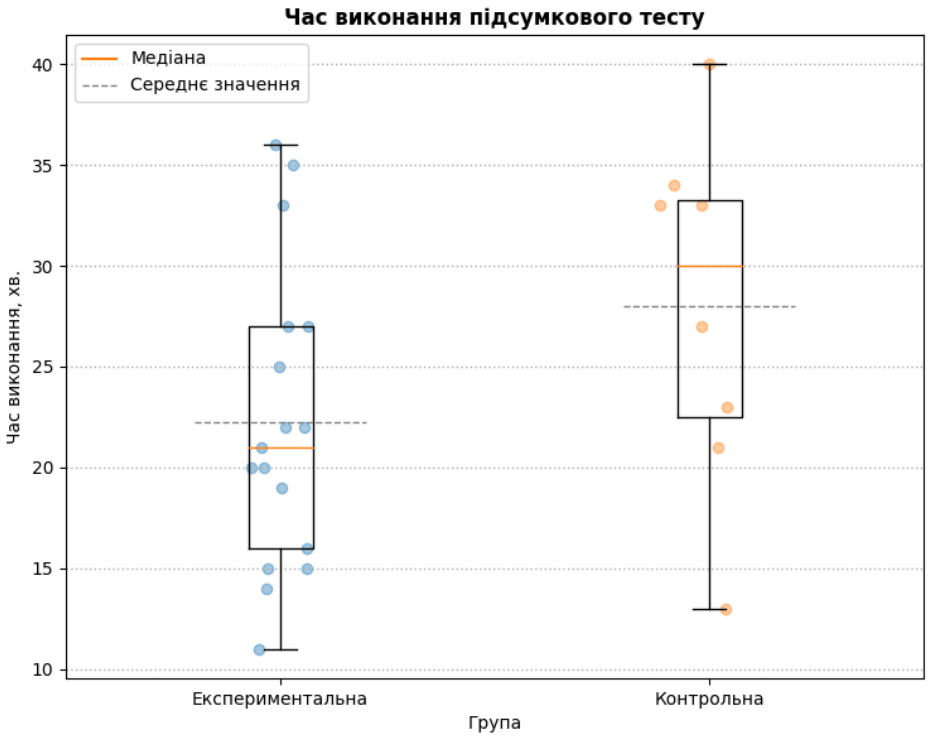


Рисунок 3.34 — Часові витрати на підсумковий тест за групами

Результати анкетування експериментальної (див. рис. 3.35) та контрольної (див. рис. 3.36).

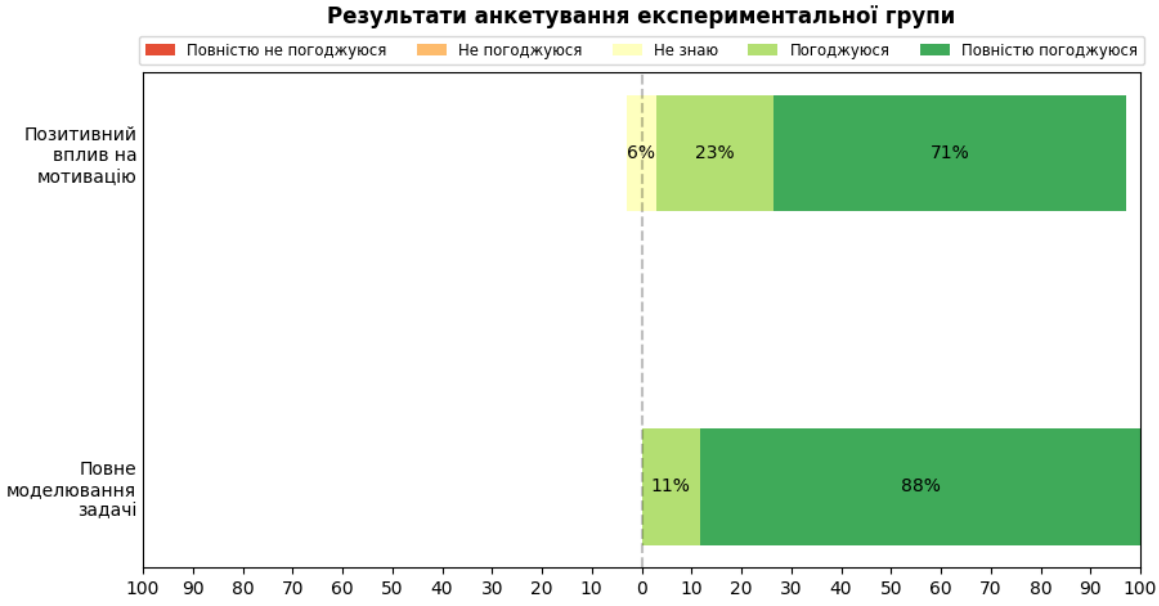


Рисунок 3.35 — Результати анкетування експериментальної групи

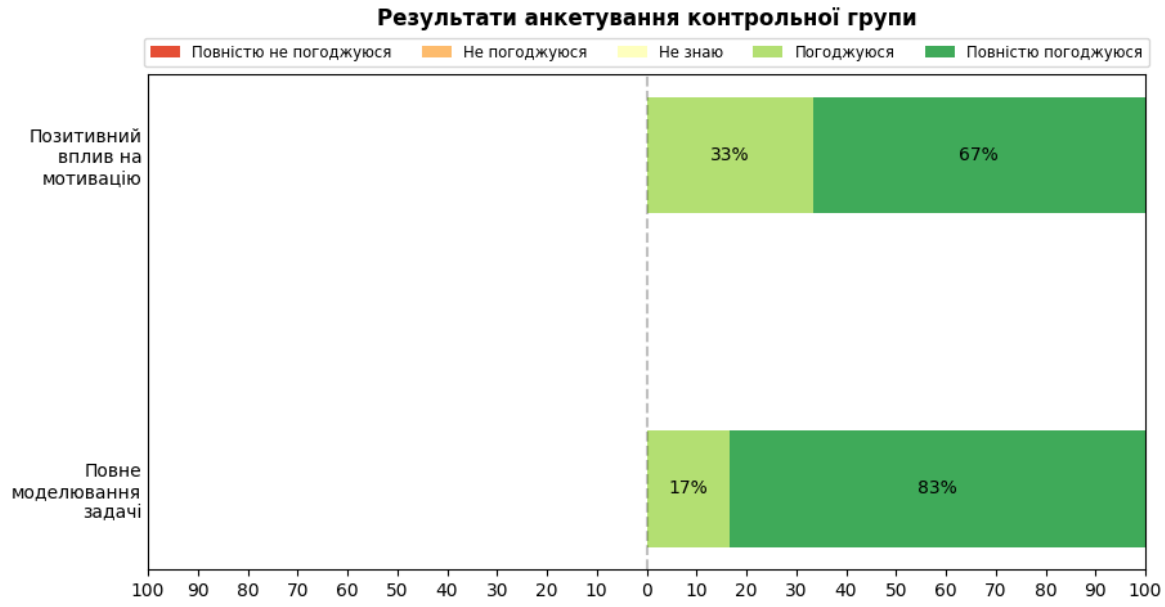


Рисунок 3.36 — Результати анкетування контрольної групи

Видно, що рівень упевненості тих, хто погоджується із запропонованими твердженнями, є вищим: 71% проти 67% за першим запитанням (про позитивний впив на мотивацію до навчання) та 88% проти 83% за другим (про повноту моделювання транспортної задачі).

За абсолютними показниками, якщо провести обчислення середніх значень за відповідями на кожне запитання (див. рис. 3.37, 3.38). Учасники високо оцінили вплив на мотивацію до навчання: 1.667 на інтервалі від −2 до 2 в обох групах. Точність співпадіння середніх можна пояснити невеликою кількістю даних (особливо у випадку контрольної групи, де анкетування пройшло 6 осіб — див. рис. 3.28) та дискретністю оцінок, через що маємо невелику можливу варіативність значень: −2, −1, 0, 1. 2.

Студенти високо оцінили повноту моделювання процесу розв’язання транспортних задач обома тренажерами: 1.867 в експериментальній групі та 1.833 у контрольній на інтервалі від −2 до 2.

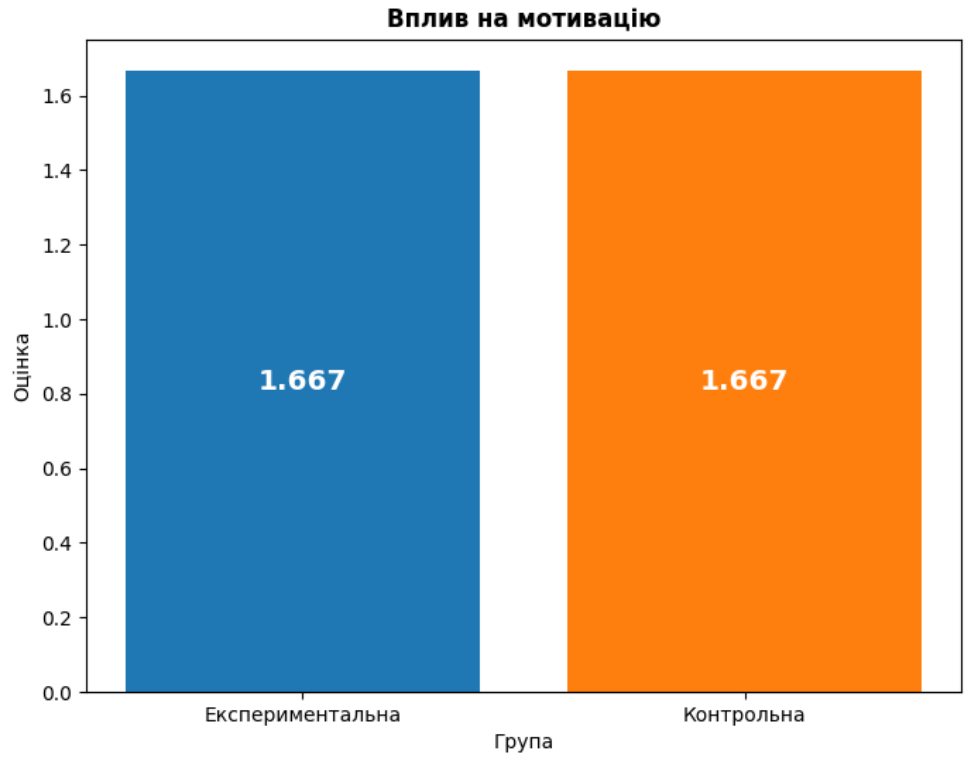


Рисунок 3.37 — Середні оцінки впливу на мотивацію

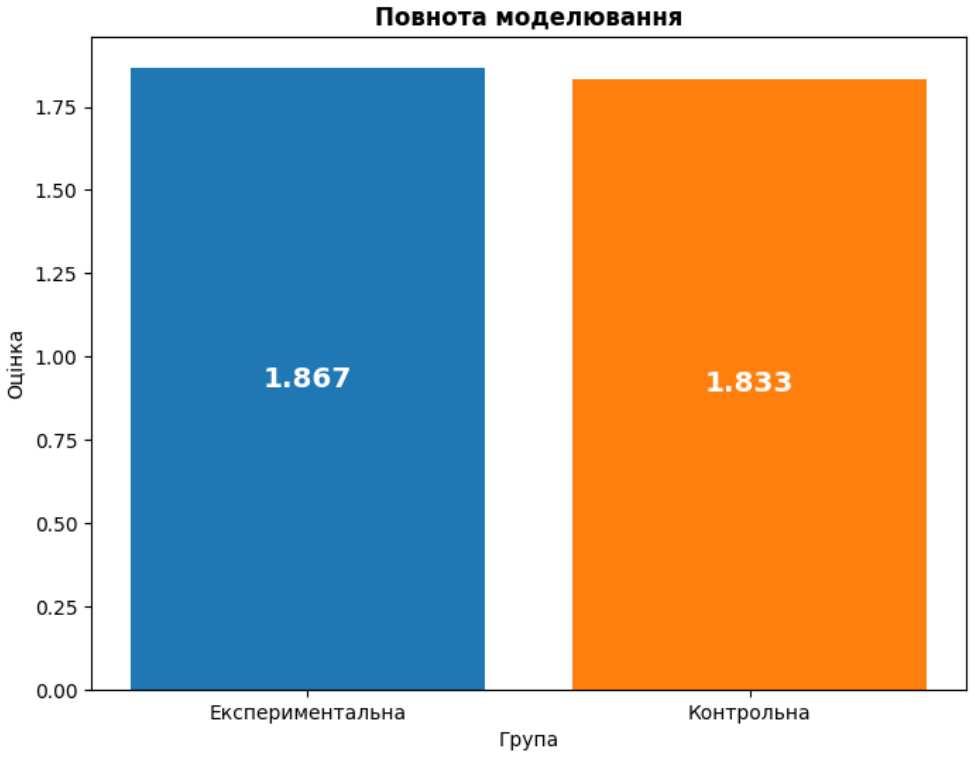


Рисунок 3.38 — Середні оцінки повноти моделювання

Завдяки використанню переваг когнітивізму тренажери в обох групах сприяли засвоєнню знань із розв’язання транспортних задач.

Експериментальна група, що працювала з тренажером на основі скінченного автомата з додатковою пам’яттю, продемонструвала дещо кращі показники. Є тенденція до виявлення статистично значущих різниць за приростом оцінки та часовими витратами.

Результати анкетування виявилися приблизно однаковими з кількох причин. По-перше, різниця в розмірах вибірок: зазвичай за більшої кількості точок даних зменшується дисперсія, що позитивно впливає на точність та надійність висновків; якщо інформації мало — маємо нестійкі результати, що чутливі до викидів. По-друге, недосконаліть дизайну експерименту: учасники не мали змогу порівнювати тренажери. За умов ізоляції студенти вважали, що працюють із корисним для навчання продуктом, не маючи уявлення про існування кращої або гіршої альтернативи. По-третє, вплив на мотивацію в реальності, ймовірно, не однаковий, як на рисунку 3.37, про що може свідчити різна конверсія етапу анкетування (див. рис. 3.28): 88.235% у експериментальної групи проти 75% у контрольної. По-четверте, повноту моделювання транспортної задачі оцінити студенту достатньо складно, оскільки він лише вивчає матеріал та не володіє більш повним уявленням про тему, як викладач.

Абсолютний приріст оцінки експериментальної групи виявився меншим за той, що в контрольній, оскільки перша мала незначну перевагу в рівні знань, що обумовило менший простір для покращення. Медіана оцінки за поточний тест для експериментальної групи $M_1 = 77.75\%$, а контрольної — $M_2 = 66.9\%$. Різниця між медіанами: 10.85% (відсотки — одиниця вимірювання оцінки). Порівнюємо саме медіани, оскільки вони менш чутливі до викидів. Різниця не є статистично значущою за результатами застосування U-критерію, утім достатньо велика, щоб негативно вплинути на приріст оцінки експериментальної групи. За умови приблизної рівності медіан приріст на користь основного тренажера на рисунку 3.31 був би більшим.

Аналогічно для економії часу: медіана експериментальної групи $M_1 = 37.5$ хв., а контрольної — $M_2 = 46$ хв. Різниця між ними: 8.5 хв. За приблизної рівності медіан економія часу на користь основного тренажера на рисунку 3.32 була б вищою.

Дане припущення щодо наявності переваг на користь експериментальної групи на етапі поточного тестування підтверджується графіками абсолютних показників (див. рис. 3.29, 3.30). Експеримент було продовжено, оскільки перевірки за t-критерієм Стьюдента (для часових витрат, що розподілені нормально) та U-критерієм Манна-Уітні (для оцінок; їхній розподіл для експериментальної групи відмінний від нормального) не виявляли статистично значущих відмінностей між отриманими вибірками.

Також на результати експерименту могла вплинути деяка схожість завдань альтернативного тренажера із умістом поточного та підсумкового тестування. Даний навчальний об’єкт за використання звичайної автоматної моделі та переваг генерації за своєю суттю є динамічним тестом.

Крім того у контрольній групі наприкінці експерименту могли лишитися найбільш відповідальні студенти, навчальні досягнення яких зазвичай є кращими.

ВИСНОВКИ

У рамках виконання кваліфікаційної роботи проведений ґрунтовний аналіз останніх наукових публікацій, присвячених створенню інтерактивних навчальних матеріалів. Автори праць в основному спрямовували свої зусилля на дослідження ефективності технологій віртуальної та доповненої реальності для розроблення навчальних симуляторів. Науковці вбачали значний потенціал для освіти у використанні сучасних інтерфейсів взаємодії людини з комп'ютером. Незважаючи на підвищений інтерес до цього нового перспективного напряму розвитку технологій електронного навчання, уже відомі методи, що спираються на використання віртуальних тренажерів та інтерактивних підручників, не втрачають своєї актуальності. Їхня інтеграція до освітнього процесу ще продовжується.

Використання запропонованої автоматної моделі віртуального тренажера з додатковою пам'яттю сприятиме збільшенню гнучкості у формуванні завдань, що матиме позитивний вплив на ефективність освітнього процесу та швидкість впровадження. Механізм додаткової пам'яті дозволяє проводити моделювання значно ширшого кола задач, що створює сприятливе підґрунтя для появи більшої кількості інтерактивних навчальних об'єктів.

Для практичного застосування розробленої автоматної моделі створена графічна нотація діаграми станів. Насамперед вона слугує інструментом формалізації логіки функціонування тренажера. До того ж використовувані позначення дозволять порозумітися математикам і розробникам під час спільної роботи над створенням навчальних об'єктів математичного спрямування.

Розроблена графічна нотація для візуалізації n-арного дерева з керувальними вузлами. Її застосування дозволяє сформувати уявлення про процес оцінювання навчальних досягнень студентів, що працюють із тренажером, створеного на основі нової технології.

Програмна реалізація технології являє собою фреймворк, що дозволяє розробляти віртуальні навчальні тренажери з гнучкою логікою формування завдань для користувача. Розробка лишатиметься актуальною тривалий час завдяки використанню сучасних технологій у її основі. Створювані навчальні об’єкти інтегруються до платформ Сумського державного університету, Mіх та Lectur.ED.

Із застосуванням запропонованого фреймворку створено віртуальний тренажер на тему «Транспортні задачі», що був упроваджений для подальшого використання в рамках дисципліни «Математичні методи дослідження операцій» (акт впровадження від 13.05.2024 р.). Навчальний об’єкт слугуватиме прикладом для створення інших тренажерів за цією технологією.

Для проведення експериментального дослідження розроблений альтернативний тренажер із використанням фреймворку jTrainer. Незважаючи на обмеженість моделі, у подальшому він може бути використаний як інструмент генерації транспортних задач, що відповідають певним умовам. Їх можна вводити до процедури контролю знань студентів та розроблення методичних указівок. Це гарне рішення, оскільки дозволяє зекономити час викладача на створення транспортних задач, у яких спостерігається певна особлива ситуація, яку достатньо складно отримати вручну, наприклад, щоб цикл переходу містив самоперетини.

Інші матеріали експерименту також можуть бути корисними. За завданнями поточного та підсумкового тесту можна проводити контролі знань із теми «Транспортні задачі». Інструкції до тренажерів залишаються актуальними.

За результатами освітнього експерименту тренажер на основі нової технології продемонстрував позитивний вплив на результати тестування. Абсолютний приріст за кількістю балів склав у середньому 9.294%, де відсотки — одиниця вимірювання. Економія часу в середньому склала 14.529 хв. Вплив на мотивацію до навчання учасники оцінили на 1.667 на інтервалі від −2 до 2. Також студенти високо оцінили повноту моделювання транспортної задачі: 1.867 на інтервалі від −2 до 2. Утім наявної кількості учасників у групах не вистачило для демонстрації статистично значущих переваг. Зумовлена формою участі на вільних засадах та

нестабільністю ситуації в країні різна конверсія на етапі поточного тестування (68% для контрольної групи та 32% для експериментальної) призвела до опрацювання на подальших етапах груп, що хоч і не відрізняються статистично значуще за U-критерієм Манна-Уітні, утім мають різні структури розподілів. Їхні статистики достатньо суттєво відрізняються між собою. Наприклад, медіана кількості балів за поточне тестування експериментальної групи (77.75%) вища за ту, що в контрольної (66.9%), на 10.85%, де відсотки — одиниця вимірювання балів. Аналогічно з медіанами часових витрат: 37.5 хв. проти 46 хв. відповідно. Експериментальна група мала кращі результати на початку, а тому простір для їхнього покращення завдяки використанню тренажера був суттєво меншим. Це підтверджується графіками з абсолютними показниками. Середня оцінка за виконання підсумкового тесту експериментальною групою склала 89.247%, а контрольної — 86.262%. Середній час виконання для експериментальної групи дорівнює 22.235 хв., а контрольної складає 28 хв.

Для доведення більш суттєвих переваг основного тренажера над альтернативним у майбутньому потрібен додатковий експеримент із забезпеченням більшої кількості учасників та, відповідно, рівномірності груп. Для цього варто відмовитися від припущення про рівномірність відвідування занять і реалізувати процедуру реєстрації на експеримент.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шовкопляс О.А., Сивоконь В.В. Інтерактивний тренажер з теми «Транспортні задачі» // Наука і техніка сьогодні. 2023. № 13(27).
2. Сивоконь В.В., Шовкопляс О.А. Програмна реалізація інтерактивного практичного завдання з теми «Транспортна задача» // Інформатика, математика, автоматика. Суми: Сумський державний університет, 2023. С. 117–118.
3. Сивоконь В.В., Шовкопляс О.А. Скінченний автомат із додатковою пам’яттю як інструмент проєктування віртуальних тренажерів // Інформатика, математика, автоматика. Суми: Сумський державний університет, 2024. С. 93–94.
4. Sivokon V., Shovkoplias O. Framework for developing interactive trainers // Information Technology and Implementation (IT&Is-2023). Kyiv: Каравела, 2023. С. 257–258.
5. Ботузова Ю., Осіпова В. Інтерактивні тренажери під час вивчення теми «Метод математичної індукції». 2022.
6. Ouh E.L., Kok Siew Gan B., Lo D. ITSS: Interactive Web-Based Authoring and Playback Integrated Environment for Programming Tutorials // Proceedings - International Conference on Software Engineering. 2022.
7. Xhako A. et al. Gamified experiences using 360° photography: A methodology for creating gamified learning experiences in 360° virtual environments // ACM International Conference Proceeding Series. 2023.
8. Ericson B.J., Miller B.N. Runestone: A platform for free, on-line, and interactive ebooks // SIGCSE 2020 - Proceedings of the 51st ACM Technical Symposium on Computer Science Education. 2020.
9. Larraga-García B. et al. Design and Development of an Interactive Web-Based Simulator for Trauma Training: A Pilot Study // J. Med. Syst. 2021. Vol 45, №

11.

10. Bohne T. et al. Gamification Intensity in Web-Based Virtual Training Environments and Its Effect on Learning // IEEE Trans. Learn. Technol. 2023. Vol 16, № 5.
11. Garcia-Carballeira F. et al. WepSIM: An Online Interactive Educational Simulator Integrating Microdesign, Microprogramming, and Assembly Language Programming // IEEE Trans. Learn. Technol. 2020. Vol 13, № 1.
12. Ramaha N.T.A., Abdellatef H., Karas I.R. An Intelligent Tutoring System To Maintain The Students' Motivation // International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives. 2022. Vol 48, № 4/W3-2022.
13. De Santo A. et al. Promoting Computational Thinking Skills in Non-Computer-Science Students: Gamifying Computational Notebooks to Increase Student Engagement // IEEE Trans. Learn. Technol. 2022. Vol 15, № 3.
14. Chong K.E. et al. Flipped-classroom with interactive videos in first year undergraduate physics course in Hong Kong. 2019.
15. Martin-Villalba C., Urquia A. An Approach to Develop Collaborative Virtual Labs in Modelica // IEEE Access. 2022. Vol 10.
16. Marcen B., Fošner M., Knežević N. Advanced Interactive Mathematical Tasks (DLO) for Moodle // Int. J. Interact. Mob. Technol. 2022. Vol 16, № 11.
17. Seemann E. MathAuthor: Authoring interactive math exercises for the Web // CSEDU 2016 - Proceedings of the 8th International Conference on Computer Supported Education. 2016. Vol 1.
18. Окрепка Г. Використання віртуальних тренажерів PhET–interactive simulation при викладанні загальної хімії фармацевтам у закладах вищої освіти // Освітологічний дискурс. Borys Grinchenko Kyiv University, 2020. № 3. С. 206–219.
19. Assaf H., Vacca A. Virtual reality tool to support fluid power curriculum // Comput. Appl. Eng. Educ. 2023. Vol 31, № 5.

20. Shang X., Kallmann M., Arif A.S. Effects of correctness and suggestive feedback on learning with an autonomous virtual trainer // International Conference on Intelligent User Interfaces, Proceedings IUI. 2019.
21. Gulec U. et al. Applying virtual reality to teach the software development process to novice software engineers // IET Softw. 2021. Vol 15, № 6.
22. Vogt P. et al. Designing and Evaluating a Virtual Reality Training for Paramedics to Practice Triage in Complex Situations // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2023. Vol 14027 LNCS.
23. Cassola F. et al. Design and Evaluation of a Choreography-Based Virtual Reality Authoring Tool for Experiential Learning in Industrial Training // IEEE Trans. Learn. Technol. 2022. Vol 15, № 5.
24. Shen H. Exploring the Potential of Virtual Reality in Table Tennis Training through Hybrid Motion Capture // Comput. Aided. Des. Appl. 2023. Vol 20, № S14.
25. Feinberg R.R. et al. ZenVR: Design Evaluation of a Virtual Reality Learning System for Meditation // Conference on Human Factors in Computing Systems - Proceedings. 2022.
26. Al-Amri S. et al. A framework for designing interactive mobile training course content using augmented reality // Multimed. Tools Appl. 2023. Vol 82, № 20.
27. Sung N.J. et al. Real-time augmented reality physics simulator for education // Appl. Sci. 2019. Vol 9, № 19.
28. Yang K.H. et al. Promoting Students' Math Learning Performance and Engagement: A Help-seeking Mechanism-based Mobile Gaming Approach // J. Internet Technol. 2022. Vol 23, № 6.
29. Кузьмич О. et al. Дослідження ефективності мобільного додатку-тренажера для вивчення англійської мови. // Comput. Technol. Educ. Sci. Prod. 2020. № 41.
30. Caruso F., Di Mascio T., Pennese M. Gamify the audiation: The Crazysquare

project // CSEDU 2019 - Proceedings of the 11th International Conference on Computer Supported Education. 2019. Vol 1.

31. Mammana M.F. et al. A Case Study of Teachers' First Designs Using ASYMPTOTE: A Tool for Teaching and Learning Mathematics Online // Communications in Computer and Information Science. 2023. Vol 1779 CCIS.
32. Corral Abad E. et al. Improving the learning of engineering students with interactive teaching applications // Comput. Appl. Eng. Educ. 2021. Vol 29, № 6.
33. Atin S., Syakuran R.A., Afrianto I. Implementation of Gamification in Mathematics m-Learning Application to Creating Student Engagement // Int. J. Adv. Comput. Sci. Appl. 2022. Vol 13, № 7.
34. Sánchez M.J.N. et al. Adaptive smart TV as asocial language learning platform // 16th International Conference on Cognition and Exploratory Learning in Digital Age, CELDA 2019. 2019.
35. Juera L.C. Digitalizing skills development using simulation-based mobile (SiM) learning application // J. Comput. Educ. 2022.
36. Rosmiati U., Siregar N. Promoting Prezi-PowerPoint presentation in mathematics learning: The development of interactive multimedia by using ADDIE model // Journal of Physics: Conference Series. 2021. Vol 1957, № 1.
37. Ahn K. A teaching model for undergraduate students // Int. J. High. Educ. 2019. Vol 8, № 3.
38. Михалевич В.М., Гнідунець В.О., Ланова В.С. Навчальний Maple-тренажер із застосування диференціала функції однієї змінної до наближених обчислень // L Науково-технічна конференція підрозділів ВНТУ. Вінниця: Вінницький національний технічний університет, 2021.
39. Ольховська О.В. et al. Технологія програмної реалізації тренажеру з теми «Системи числення, арифметичні операції в різних системах числення» дисципліни «Архітектура обчислювальних систем» // Вісник Херсонського національного технічного університету. Kherson National

Technical University, 2023. № 3(82). С. 65–73.

40. Farooq U. et al. The impact of design on improved learning in virtual worlds: an experimental study // Multimed. Tools Appl. 2022. Vol 81, № 13.
41. Валєєв Р.Г. Використання віртуальних діалогових тренажерів у професійній підготовці поліцейських // Психологічні та педагогічні проблеми професійної освіти та патріотичного виховання персоналу системи МВС України. Харків, 2018. С. 70–72.
42. Герганов Л.Д., Ягупов В.В. Досвід і перспективи застосування сучасного тренажерного обладнання у навчально-тренувальних центрах морського профілю // Inf. Technol. Learn. Tools. 2020. Vol 78, № 4.
43. Filatova Z., Galyamova E., Burkhanova Y. Digital Learning Tools and Devices for the Implementation of an Electronic Educational Resource // Lecture Notes in Networks and Systems. 2023. Vol 509.
44. Бабенко Д.В., Доценко Н.А. Методика створення навчальних комп’ютерних інтерактивних тренажерів для вивчення загальнотехнічних дисциплін в умовах інформаційно-освітнього середовища. Миколаїв: МНАУ, 2019.
45. Доценко Н. Застосування навчальних комп’ютерних інтерактивних тренажерів здобувачами вищої освіти інженерних спеціальностей в умовах інформаційно-освітнього середовища // Педагогічні науки: теорія, історія, інноваційні технології. 2018. № 2(76). С. 118–128.
46. Nur Hidayat W. et al. The effectiveness of interactive digital evaluation training for improving teacher skills in the covid-19 pandemic period // 4th International Conference on Vocational Education and Training, ICOVET 2020. 2020.
47. Михалевич В.М., Тютюнник О.І. Інтелектуальні навчальні тренажери розв’язування задач лінійного програмування як елемент інформаційно-комунікаційних технологій навчання // Теорія та методика електронного навчання. 2012. № III. С. 195–199.
48. Pedro L.Z. et al. Does gamification work for boys and girls? An exploratory

study with a virtual learning environment // Proceedings of the ACM Symposium on Applied Computing. 2015. Vol 13-17-Apri.

49. De-Marcos L. et al. Gamifying massive online courses: Effects on the social networks and course completion rates // Appl. Sci. 2020. Vol 10, № 20.
50. Остапенко В.О. Методичні положення з побудови комп'ютерних віртуальних тренажерів в системах дистанційного навчання // Штучний інтелект. 2019. № 3–4. С. 114–125.
51. Online Calculator: Potential Method [Online]. URL: https://linprog.com/en/main-potentials (accessed: 25.12.2023).
52. Кучма М.І. Математичне програмування: приклади і задачі. 2007.
53. Лавров Є.А. et al. Математичні методи дослідження операцій. Суми: Сумський державний університет, 2017.
54. Шовкопляс О.А. Побудова математичних моделей проблемних ситуацій: транспортні задачі [Online] // Mix. URL: https://mix.sumdu.edu.ua/textbooks/77942/418447/index.html (accessed: 06.05.2024).
55. Іксанов О.М., Шевченко В.І. Транспортна задача, її властивості та методи розв'язування (курс «Дослідження операцій»). Київ: ТВіМС, 2010.
56. Romesh T. Should you use Bun or Node.js or Deno in 2024? [Online] // Bits and Pieces. 2023. URL: https://blog.bitsrc.io/should-you-use-bun-or-node-js-or-deno-in-2024-b7c21da085ba (accessed: 05.05.2024).
57. Greif S., Burel E. State of JavaScript 2022: Other Tools [Online]. 2023.
58. Sardhara K. PNPM Vs. NPM Vs. Yarn: Which Package Manager is Right for You? [Online] // DhiWise. 2024. URL: https://www.dhiwise.com/post/pnpm-vs-npm-vs-yarn-which-javascript-package-manager (accessed: 05.05.2024).
59. npm, pnpm, yarn, yarnpkg [Online] // Tag Trends. URL: http://sotagtrends.com/?tags=npm+pnpm+yarn+yarnpkg (accessed: 05.05.2024).
60. Hoekstra R. Angular vs React vs Vue.js: Comparison of Frameworks [Online] //

TechMagic. 2022. URL: https://www.techmagic.co/blog/reactjs-vs-angular-vs-vuejs/ (accessed: 06.05.2024).

61. reactjs, angular, vue.js [Online] // Tag Trends. URL: http://sotagtrends.com/?tags=reactjs+angular+vue.js (accessed: 06.05.2024).
62. Singh V. React Native/React: Babel Configuration [Online] // Medium. 2023. URL: https://thevinaysingh.medium.com/babel-configuration-in-react-native-ed7437fd616c (accessed: 06.05.2024).
63. Eaton P. Benchmarking esbuild, swc, tsc, and babel for React/JSX projects [Online] // DataStation. 2021. URL: https://datastation.multiprocess.io/blog/2021-11-13-benchmarking-esbuild-swc-typescript-babel.html (accessed: 06.05.2024).
64. IndiDev. Webpack Demystified: A Beginner's Guide and Pros & Cons [Online] // Medium. 2023. URL: https://medium.com/@IndiDev/webpack-demystified-a-beginners-guide-and-pros-cons-2c10dc71e85 (accessed: 06.05.2024).
65. esbuild — An extremely fast bundler for the web [Online]. URL: https://esbuild.github.io/ (accessed: 06.05.2024).
66. Buckler C. An Introduction to the esbuild Bundler [Online] // SitePoint. 2023. URL: https://www.sitepoint.com/esbuild-introduction/ (accessed: 06.05.2024).
67. El Banna A. Webpack vs Parcel: Which Bundler Should You Use? [Online] // LinkedIn. 2023. URL: https://www.linkedin.com/pulse/webpack-vs-parcel-which-bundler-should-you-use-ahmed-el-banna/ (accessed: 06.05.2024).
68. Chen H.J. State of CSS 2023: Other Tools [Online]. 2023. URL: https://2023.stateofcss.com/en-US/other-tools/ (accessed: 06.05.2024).
69. Sass: Values [Online]. URL: https://sass-lang.com/documentation/values/ (accessed: 06.05.2024).
70. SASS vs LESS vs SCSS – That are the differences [Online] // IONOS. 2020. URL: https://www.ionos.com/digitalguide/websites/web-development/sass/ (accessed: 06.05.2024).
71. next vs react-scripts vs vite [Online] // npm trends. URL:

https://npmtrends.com/next-vs-react-scripts-vs-vite (accessed: 06.05.2024).

72. Architecture: Next.js Compiler [Online] // Next.js. URL: https://nextjs.org/docs/architecture/nextjs-compiler (accessed: 06.05.2024).
73. Software Engineer Z. What are the Disadvantages of Next.js? [Online] // Medium. 2023. URL: https://medium.com/@devdo/what-are-the-disadvantages-of-next-js-b99201efda83 (accessed: 06.05.2024).
74. Dependency Pre-Bundling [Online] // Vite. URL: https://vitejs.dev/guide/dep-pre-bundling (accessed: 06.05.2024).
75. Why Vite [Online] // Vite. URL: https://vitejs.dev/guide/why#why-not-bundle-with-esbuild (accessed: 06.05.2024).
76. Davidson T. What is Vite, And Why Is It Awesome? [Online] // Clean Commit. 2023. URL: https://cleancommit.io/blog/what-is-vite/ (accessed: 06.05.2024).
77. adm-zip vs archiver vs jszip [Online] // npm trends. URL: https://npmtrends.com/adm-zip-vs-archiver-vs-jszip (accessed: 06.05.2024).
78. Building for Production [Online] // Vite. URL: https://vitejs.dev/guide/build (accessed: 06.05.2024).
79. es modules [Online] // Can I use. URL: https://caniuse.com/?search=es modules (accessed: 06.05.2024).
80. «Can I use» usage table [Online] // Can I use. 2024. URL: http://web.archive.org/web/20240406113615/https://caniuse.com/usage-table (accessed: 06.05.2024).
81. DisLido. Why the default browser support version of vite is higher than the features needed? [Online] // GitHub. 2023. URL: https://github.com/vitejs/vite/discussions/12116 (accessed: 06.05.2024).
82. Dhanani H. A Complete Guide to React State Management [Online] // eTatvaSoft. 2023. URL: https://www.etatvasoft.com/blog/react-state-management/ (accessed: 06.05.2024).
83. @hookstate/core vs @rematch/core vs jotai vs mobx vs recoil vs redux vs zustand [Online] // npm trends. URL: https://npmtrends.com/@hookstate/core-

vs-@rematch/core-vs-jotai-vs-mobx-vs-recoil-vs-redux-vs-zustand (accessed: 06.05.2024).

84. Israel. Redux vs Zustand vs Context API: Their Pros, Cons, and Usage [Online] // Bootcamp. 2023. URL: https://bootcamp.uxdesign.cc/redux-vs-zustand-vs-context-api-their-pros-cons-and-usage-d3bcbb79ab6a (accessed: 06.05.2024).
85. Chen H.J. State of CSS 2023: CSS Frameworks [Online]. 2023. URL: https://2023.stateofcss.com/en-US/css-frameworks/ (accessed: 06.05.2024).
86. Why React-Bootstrap? [Online] // React Bootstrap. URL: https://react-bootstrap.netlify.app/docs/getting-started/why-react-bootstrap/ (accessed: 06.05.2024).
87. SVG animation with SMIL [Online] // MDN. 2024. URL: https://developer.mozilla.org/en-US/docs/Web/SVG/SVG_animation_with_SMIL (accessed: 06.05.2024).
88. Kamunya T. 13 Top CSS Animation Libraries for Stunning Web Design Projects [Online] // Geekflare. 2023. URL: https://geekflare.com/top-css-animation-libraries/ (accessed: 06.05.2024).
89. Ikechi F. Comparing the best React animation libraries for 2023 [Online] // LogRocket Blog. 2023. URL: https://blog.logrocket.com/best-react-animation-libraries/ (accessed: 06.05.2024).
90. @react-spring/web vs animejs vs framer-motion vs gsap vs react-move vs react-reveal vs react-transition-group vs remotion [Online] // npm trends. URL: https://npmtrends.com/@react-spring/web-vs-animejs-vs-framer-motion-vs-gsap-vs-react-move-vs-react-reveal-vs-react-transition-group-vs-remotion (accessed: 06.05.2024).
91. classnames vs clsx [Online] // npm trends. URL: https://npmtrends.com/classnames-vs-clsx (accessed: 06.05.2024).
92. Nawo A. You don’t know the classNames library [Online] // Arek Nawo Ideas. 2021. URL: https://areknawo.com/you-dont-know-the-classnames-library/ (accessed: 06.05.2024).

93. Run results for: Classnames vs CLSX vs Alternatives [Online] // MeasureThat.net. 2024. URL: https://www.measurethat.net/Benchmarks/ShowResult/517880 (accessed: 06.05.2024).
94. date-fns vs dayjs [Online] // npm trends. URL: https://npmtrends.com/date-fns-vs-dayjs (accessed: 06.05.2024).
95. Shah S. Date fns vs. Dayjs: A Developer's Guide to Making the Right Choice [Online] // DhiWise. 2024. URL: https://www.dhiwise.com/post/date-fns-vs-dayjs-the-battle-of-javascript-date-libraries (accessed: 06.05.2024).
96. MathML [Online] // MDN. 2023. URL: https://developer.mozilla.org/en-US/docs/Web/MathML (accessed: 06.05.2024).
97. MathML [Online] // Can I use. URL: http://web.archive.org/web/20240331120409/https://caniuse.com/mathml (accessed: 06.05.2024).
98. katex vs mathjax [Online] // npm trends. URL: https://npmtrends.com/katex-vs-mathjax (accessed: 06.05.2024).
99. react-beautiful-dnd vs react-dnd vs react-draggable [Online] // npm trends. URL: https://npmtrends.com/react-beautiful-dnd-vs-react-dnd-vs-react-draggable (accessed: 06.05.2024).
100. RAND (функція RAND) [Online] // Підтримка від Microsoft. URL: https://support.microsoft.com/uk-ua/office/rand-функція-rand-4cbfa695-8869-4788-8d90-021ea9f5be73 (accessed: 06.05.2024).
101. Тран А. 40+ найкращих прикладів шкали Лайкерта [Online] // AhaSlides. 2024. URL: https://ahaslides.com/uk/blog/likert-scale-examples/ (accessed: 06.05.2024).
102. Повна підтримка ES5 та Flexbox [Online] // Browserslist. URL: https://browsersl.ist/#q=fully+supports+es5+and+fully+supports+flexbox+and+not+dead (accessed: 06.05.2024).
103. Shapiro Wilk Test [Online] // Statistics Kingdom. URL:

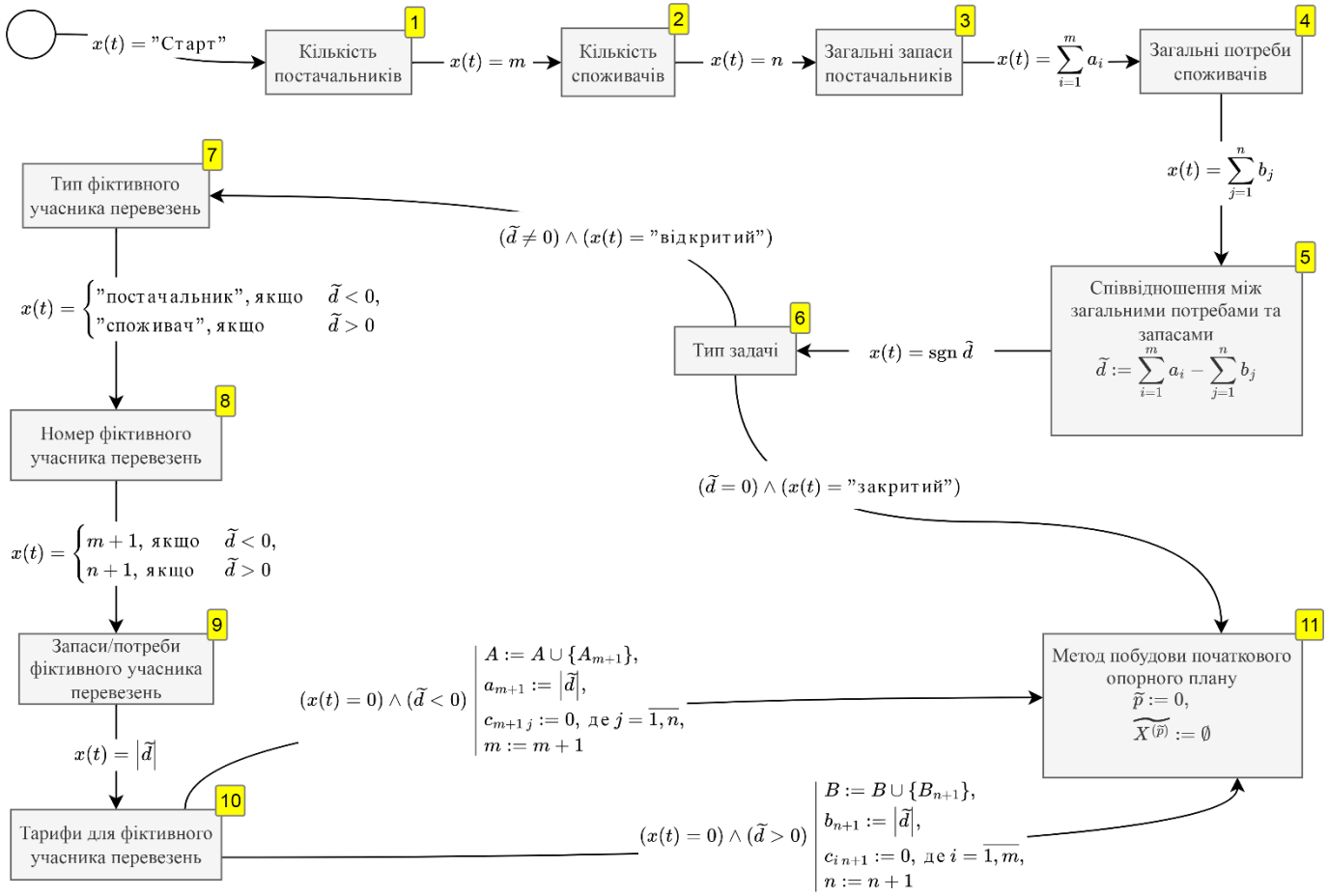
https://www.statskingdom.com/doc_shapiro_wilk.html (accessed: 06.05.2024).

104. LaFromboise L. Royston Shapiro-Wilk Test p-value Formula — Engineering Statistics [Online] // YouTube. 2024. URL: https://www.youtube.com/watch?v=dKJL0sEyG1c&list=PLJWeb5_BSlfPDtixKAV4cuiMZNhl10rFR (accessed: 06.05.2024).
105. Shared Options [Online] // Vite. URL: https://vitejs.dev/config/shared-options (accessed: 09.05.2024).

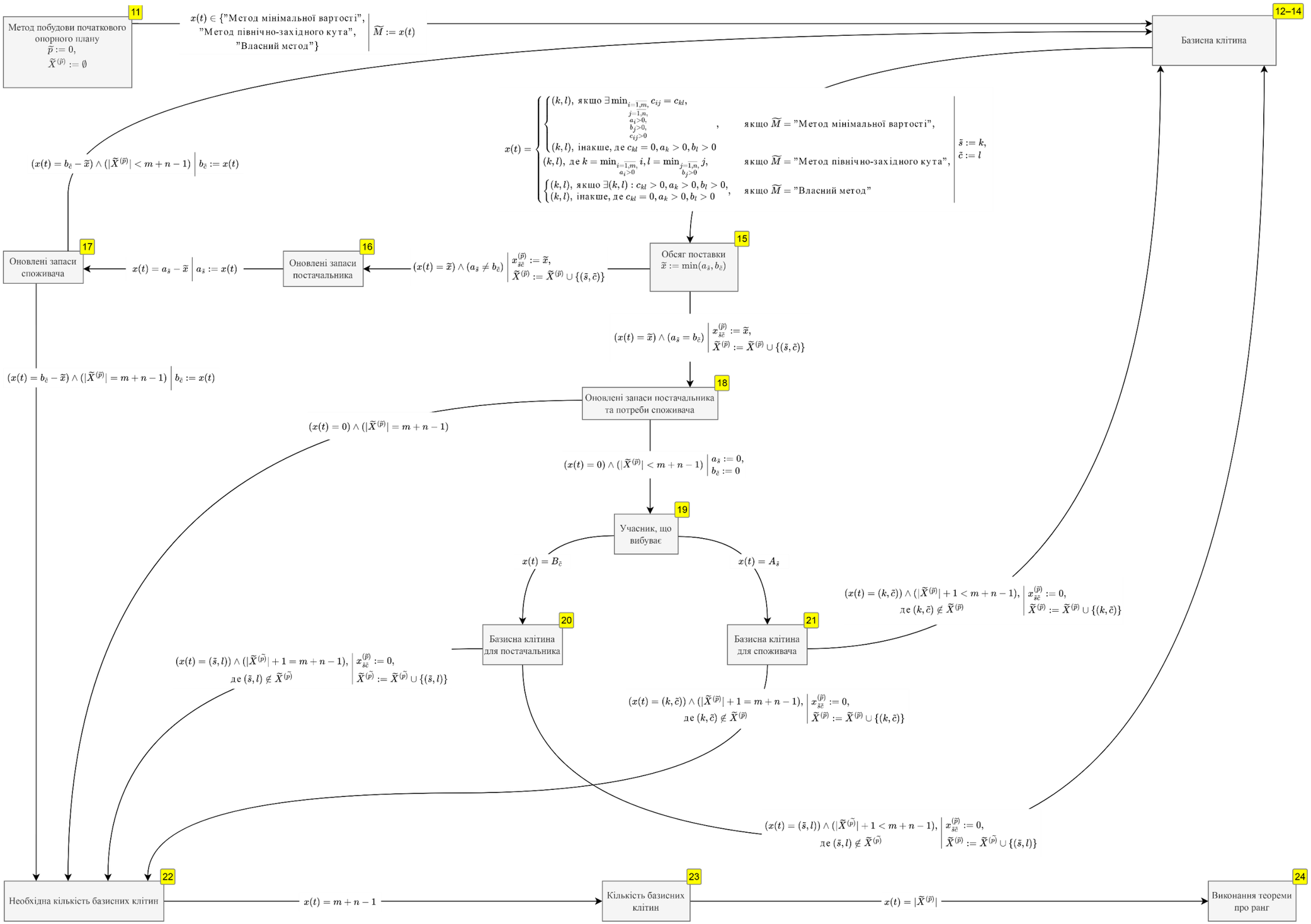
ДОДАТОК А

ДІАГРАМА СТАНІВ REACT-ТРЕНАЖЕРА З ТРАНСПОРТНИХ ЗАДАЧ

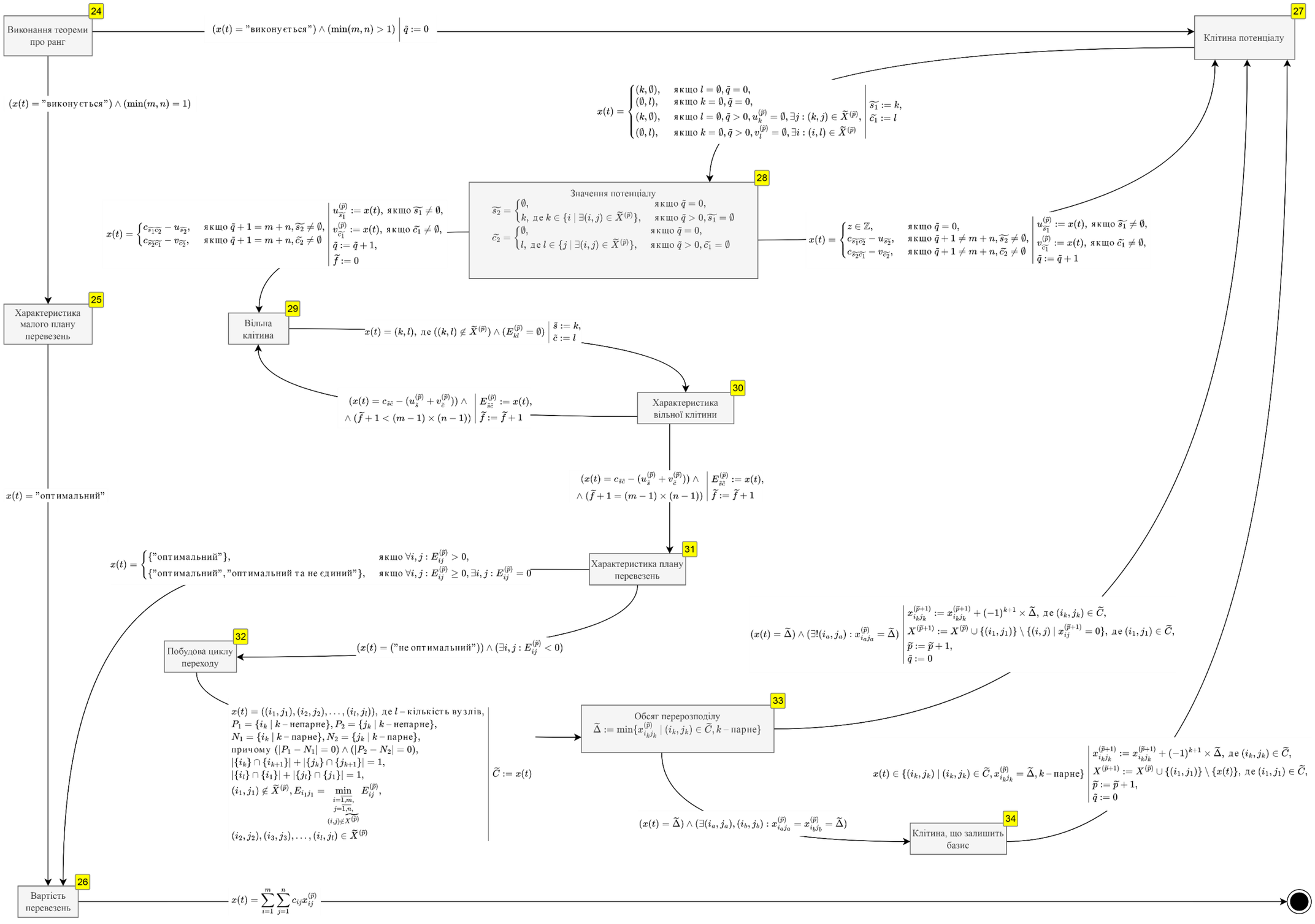
А.1 Етап 1: основні властивості



A.2 Етап 2: план перевезень



A.3 Етап 3: метод потенціалів



ДОДАТОК Б

ЗАВДАННЯ ПОТОЧНОГО ТЕСТУ

Б.1 Основні властивості

1. Введіть значення параметрів запропонованої транспортної задачі.

Споживачі / Постачальники		B_1	B_2	B_3	B_4
		110	150	170	140
A_1	240	5	5	20	15
A_2	190	20	15	5	5
A_3	160	20	10	25	5

Кількість постачальників: _____.

Кількість споживачів: _____.

Загальний обсяг запасів постачальників: _____.

Загальний обсяг потреб споживачів: _____.

2. Визначте тип кожної транспортної задачі: відкритий чи закритий.

1. Транспортна задача відкритого типу
2. Транспортна задача закритого типу

1.

Споживачі / Постачальники		B_1	B_2	B_3
		20	30	10
A_1	50	5	20	5
A_2	20	15	5	20

2.

Споживачі / Постачальники		B_1	B_2	B_3	B_4
		20	80	20	30
A_1	40	15	20	5	15
A_2	60	25	10	10	5

3.

Споживачі / Постачальники		B_1	B_2	B_3
		20	40	20
A_1	60	5	10	5
A_2	30	30	10	20

4.

Споживачі / Постачальники		B_1	B_2
		30	70
A_1	10	5	20
A_2	60	20	15
A_3	30	10	25

5.

Споживачі / Постачальники		B_1	B_2
		40	60
A_1	70	5	15
A_2	30	20	10

3. Побудуйте правильні твердження стосовно приведення транспортної задачі до закритого типу.

Якщо загальний обсяг запасів постачальників перевищує загальний об'єм потреб споживачів, тоді треба _____ (ввести фіктивного постачальника / ввести фіктивного споживача / ввести фіктивного постачальника та споживача / нічого не робити).

Якщо загальний обсяг потреб споживачів перевищує загальний обсяг запасів постачальників, тоді треба _____ (ввести фіктивного постачальника / ввести фіктивного споживача / ввести фіктивного постачальника та споживача / нічого не робити).

Якщо загальний обсяг запасів постачальників дорівнює загальному обсягу потреб споживачів, тоді треба _____ (ввести фіктивного постачальника / ввести фіктивного споживача / ввести фіктивного постачальника та споживача / нічого не робити).

4. Обчисліть обсяг запасів/потреб фіктивних постачальників/споживачів у запропонованих транспортних задачах відкритого типу.

Споживачі / Постачальники		B_1	B_2	B_3	B_4
		10	30	20	?
A_1	20	5	10	25	0
A_2	50	20	15	5	0
A_3	30	10	5	20	0

Потреби b_4 фіктивного споживача B_4: _____.

Споживачі / Постачальники		B_1	B_2	B_3
		50	40	60
A_1	30	5	15	10
A_2	100	10	20	5
A_3	?	0	0	0

Запаси a_3 фіктивного постачальника A_3: _____.

Б.2 План перевезень

1. Користуючись теоремою про ранг матриці коефіцієнтів системи обмежень транспортної задачі обчисліть необхідну кількість базисних клітин.

Транспортну задачу відкритого типу, де було 5 постачальників та 3 споживачів, привели до закритого типу. Кількість базисних клітин: _____.

2. Установіть послідовність розгляду клітин тарифів транспортної таблиці за методом північно-західного кута.

Примітка. Зверніть увагу: B_5 — фіктивний споживач.

Споживачі / Постачальники		B_1	B_2	B_3	B_4	B_5
		110	200	100	100	80
A_1	150	15	5	15	10	0
A_2	200	20	25	20	10	0
A_3	240	15	5	10	15	0

— c_{11};

— c_{12};

— c_{22};

— c_{23};

— c_{33};

— c_{34};

— c_{35}.

3. Установіть послідовність розгляду клітин тарифів транспортної таблиці за методом мінімальної вартості.

Примітка. Зверніть увагу: A_3 — фіктивний постачальник.

Споживачі / Постачальники		B_1	B_2	B_3	B_4
		180	120	150	190
A_1	340	10	25	30	35
A_2	150	20	5	40	15
A_3	150	0	0	0	0

— c_{22};

— c_{11};

— c_{24};

— c_{13};

— c_{14};

— c_{34}.

4. Обчисліть обсяг поставки x_{11} за тарифом у клітині c_{11} транспортної таблиці, а також оновлене значення запасу постачальника A_1 та потреб споживача B_1.

Споживачі / Постачальники		B_1	B_2	B_3
		120	160	120
A_1	210	10 ?	25	0
A_2	190	15	25	0

Обсяг поставки x_{11}: _____.

Оновлений обсяг запасів постачальника A_1: _____.

Оновлений обсяг потреб споживача B_1: _____.

5. Обчисліть обсяг поставки x_{22} за тарифом у клітині c_{22} транспортної таблиці, а також оновлене значення запасу постачальника A_2 та потреб споживача B_2.

Споживачі / Постачальники		B_1	B_2	B_3
		~~130~~ **0**	220	20
A_1	190	10	20	0
A_2	~~180~~ **50**	5 130	10 ?	0

Обсяг поставки x_{22}: _____.

Оновлений обсяг запасів постачальника A_2: _____.

Оновлений обсяг потреб споживача B_2: _____.

6. Обчисліть обсяг поставки x_{12} за тарифом у клітині c_{12} транспортної таблиці, а також оновлене значення запасу постачальника A_1 та потреб споживача B_2.

Постачальники \ Споживачі		B_1	B_2
		150	~~220~~ **70**
A_1	220	25	20 ?
A_2	~~150~~ **0**	15	10 150

Обсяг поставки x_{12}: _____.

Оновлений обсяг запасів постачальника A_1: _____.

Оновлений обсяг потреб споживача B_2: _____.

7. Що треба зробити після виконаної поставки $x_{13} = 30$ у комірку c_{13}?

Примітка. Обраний варіант відповіді повинен відображати всі можливі дії.

Постачальники \ Споживачі		B_1	B_2	B_3
		20	50	~~30~~ **0**
A_1	~~30~~ **0**	30	10	5 30
A_2	70	25	15	10

а) Видалити з розгляду постачальника A_1.

б) Видалити з розгляду споживача B_3.

в) Видалити з розгляду постачальника A_1 або споживача B_3.

г) Видалити з розгляду постачальника A_1 та споживача B_3.

8. Що треба зробити після виконаної поставки $x_{23} = 30$ у комірку c_{23}?

Примітка. Обраний варіант відповіді повинен відображати всі можливі дії.

Споживачі / Постачальники		B_1	B_2	B_3
		~~30~~ **0**	~~40~~ **0**	~~30~~ **0**
A_1	~~20~~ **0**	15 20	5	0
A_2	~~80~~ **0**	20 10	10 40	0 30

а) Видалити з розгляду постачальника A_2.

б) Видалити з розгляду споживача B_3.

в) Видалити з розгляду постачальника A_2 або споживача B_3.

г) Видалити з розгляду постачальника A_2 та споживача B_3.

9. Що треба зробити після виконаної поставки $x_{11} = 30$ у комірку c_{11}?

Примітка. Обраний варіант відповіді повинен відображати всі можливі дії.

Споживачі / Постачальники		B_1	B_2	B_3	B_4
		~~30~~ **0**	40	20	60
A_1	~~50~~ **20**	5 30	15	10	5
A_2	100	25	5	15	20

а) Видалити з розгляду постачальника A_1.

б) Видалити з розгляду споживача B_1.

в) Видалити з розгляду постачальника A_1 або споживача B_1.

г) Видалити з розгляду постачальника A_1 та споживача B_1.

10. Що треба зробити після виконаної поставки $x_{22} = 50$ у комірку c_{22}?

Примітка. Обраний варіант відповіді повинен відображати всі можливі дії.

Споживачі / Постачальники		B_1	B_2
		40	~~60~~ **10**
A_1	20	15	25
A_2	~~50~~ **0**	20	5 50
A_3	30	30	10

а) Видалити з розгляду постачальника A_2.

б) Видалити з розгляду споживача B_2.

в) Видалити з розгляду постачальника A_2 або споживача B_2.

г) Видалити з розгляду постачальника A_2 та споживача B_2.

11. Обчисліть загальну вартість перевезень за планом, що наведений у вигляді транспортної таблиці.

Споживачі / Постачальники		B_1	B_2	B_3
		20	10	20
A_1	30	5 20	10 10	20
A_2	20	25	15 0	5 20

Загальна вартість перевезень: _____.

Б.3 Метод потенціалів

1. Оберіть правильні твердження щодо застосування методу потенціалів для розв’язання транспортної задачі.

а) Значення першого потенціалу обов’язково має дорівнювати 0.

б) Значення першого потенціалу є довільним.

в) Перший потенціал має визначатися для постачальника.

г) Перший потенціал має визначатися для споживача.

д) Починати визначення потенціалів можна з будь-якого учасника.

2. Обчисліть потенціал u_3 постачальника A_3 та v_2 споживача B_2.

Споживачі / Постачальники		B_1	B_2	B_3	U
		190	150	100	
A_1	230	5 190	5 40	20	$u_1 = 0$
A_2	150	15	25 50	20 100	
A_3	60	0	0 60	0	$u_3 = ?$
V			$v_2 = ?$		

Потенціал u_3 постачальника A_3: _____.

Потенціал v_2 споживача B_2: _____.

3. Оберіть вільні комірки запропонованої транспортної таблиці.

Споживачі / Постачальники		B_1	B_2	B_3
		170	200	160
A_1	250	25	5 200	15 50
A_2	280	10 170	20	30 110

а) c_{11}.

б) c_{12}.

в) c_{13}.

г) c_{21}.

д) c_{22}.

е) c_{23}.

4. Обчисліть значення Δ_{22} для позначеної вільної комірки транспортної таблиці.

Споживачі / Постачальники		B_1	B_2	B_3	U
		130	170	150	
A_1	240	25	5 170	0 70	$u_1 = 0$
A_2	210	15 130	25 ?	0 80	$u_2 = 0$
V		$v_1 = 15$	$v_2 = 5$	$v_3 = 0$	

Значення Δ_{22} дорівнює _____.

5. Оберіть правильні твердження стосовно наведеного плану перевезень.

Споживачі / Постачальники		B_1	B_2	B_3	U
		170	190	70	
A_1	250	20 10	10 190	0 60	$u_1 = 0$
A_2	180	10 170	25 15	0 10	$u_2 = 0$
V		$v_1 = 10$	$v_2 = 10$	$v_3 = 0$	

а) Оптимальний.

б) Не оптимальний.

в) Оптимальний та єдиний.

г) Оптимальний та не єдиний.

6. Оберіть правильні твердження стосовно наведеного плану перевезень.

Постачальники \ Споживачі		B_1	B_2	B_3	U
		200	190	220	
A_1	210	20 [15]	25 [40]	5 [210]	$u_1 = 0$
A_2	240	25 [50]	5 [190]	25 [0]	$u_2 = 20$
A_3	160	0 [150]	0 [20]	0 [10]	$u_3 = -5$
V		$v_1 = 5$	$v_2 = -15$	$v_3 = 5$	

а) Оптимальний.

б) Не оптимальний.

в) Оптимальний та єдиний.

г) Оптимальний та не єдиний.

7. Оберіть правильні твердження стосовно наведеного плану перевезень.

Постачальники \ Споживачі		B_1	B_2	B_3	U
		160	130	200	
A_1	150	5 [150]	20 [15]	5 [-10]	$u_1 = 0$
A_2	240	10 [10]	10 [130]	20 [100]	$u_2 = 5$
A_3	100	0 [10]	0 [10]	0 [100]	$u_3 = -15$
V		$v_1 = 5$	$v_2 = 5$	$v_3 = 15$	

а) Оптимальний.

б) Не оптимальний.

в) Оптимальний та єдиний.

г) Оптимальний та не єдиний.

8. Оберіть правильно побудовані цикли переходу.

1.

Споживачі / Постачальники		B_1	B_2	B_3	B_4	B_5	U
		170	110	100	160	200	
A_1	230	20 (−) 50	20 10	10 −10 (+)	20 160	0 20	$u_1 = 0$
A_2	160	15 50	5 110	15 0	25 10	0 5	$u_2 = -5$
A_3	180	20 0	20 10	10 −10	20 0	0 180	$u_3 = 0$
A_4	170	5 (+) 70	10 15	5 (−) 100	5 0	0 15	$u_4 = -15$
V		$v_1 = 20$	$v_2 = 10$	$v_3 = 20$	$v_4 = 20$	$v_5 = 0$	

2.

Споживачі / Постачальники		B_1	B_2	B_3	B_4	B_5	U
		170	110	100	160	200	
A_1	230	20 (−) 50	20 10	10 −10	20 160	0 (+) 20	$u_1 = 0$
A_2	160	15 50	5 110	15 0	25 10	0 5	$u_2 = -5$
A_3	180	20 0	20 10	10 −10 (+)	20 0	0 (−) 180	$u_3 = 0$
A_4	170	5 (+) 70	10 15	5 (−) 100	5 0	0 15	$u_4 = -15$
V		$v_1 = 20$	$v_2 = 10$	$v_3 = 20$	$v_4 = 20$	$v_5 = 0$	

3.

Споживачі / Постачальники		B_1	B_2	B_3	B_4	B_5	U
		170	110	100	160	200	
A_1	230	20 **50**	20 10	10 -10	20 **160**	0 **20**	$u_1 = 0$
A_2	160	15 **50**	5 **110**	15 0	25 10	0 5	$u_2 = -5$
A_3	180	20 0	20 10	10 -10	20 0	0 **180**	$u_3 = 0$
A_4	170	5 **70**	10 15	5 **100**	5 0	0 15	$u_4 = -15$
V		$v_1 = 20$	$v_2 = 10$	$v_3 = 20$	$v_4 = 20$	$v_5 = 0$	

4.

Споживачі / Постачальники		B_1	B_2	B_3	B_4	B_5	U
		170	110	100	160	200	
A_1	230	20 **50**	20 10	10 -10	20 **160**	0 **20**	$u_1 = 0$
A_2	160	15 **50**	5 **110**	15 0	25 10	0 5	$u_2 = -5$
A_3	180	20 0	20 10	10 -10	20 0	0 **180**	$u_3 = 0$
A_4	170	5 **70**	10 15	5 **100**	5 0	0 15	$u_4 = -15$
V		$v_1 = 20$	$v_2 = 10$	$v_3 = 20$	$v_4 = 20$	$v_5 = 0$	

5.

Споживачі / Постачальники		B_1	B_2	B_3	B_4	B_5	U
		170	110	100	160	200	
A_1	230	20 (+) 50	20 10	10 -10 (−)	20 160	0 20	$u_1 = 0$
A_2	160	15 50	5 110	15 0	25 10	0 5	$u_2 = -5$
A_3	180	20 0	20 10	10 -10	20 0	0 180	$u_3 = 0$
A_4	170	5 (−) 70	10 15	5 (+) 100	5 0	0 15	$u_4 = -15$
V		$v_1 = 20$	$v_2 = 10$	$v_3 = 20$	$v_4 = 20$	$v_5 = 0$	

6.

Споживачі / Постачальники		B_1	B_2	B_3	B_4	B_5	U
		170	110	100	160	200	
A_1	230	20 50	20 10	10 -10	20 160	0 20	$u_1 = 0$
A_2	160	15 (+) 50	5 (−) 110	15 0	25 10	0 5	$u_2 = -5$
A_3	180	20 0	20 10	10 -10	20 0	0 180	$u_3 = 0$
A_4	170	5 (−) 70	10 (+) 15	5 100	5 0	0 15	$u_4 = -15$
V		$v_1 = 20$	$v_2 = 10$	$v_3 = 20$	$v_4 = 20$	$v_5 = 0$	

9. Обчисліть обсяг перерозподілу Δ для побудованого циклу переходу.

Споживачі / Постачальники		B_1	B_2	B_3	U
		160	120	180	
A_1	190	10 15	15 (+) 10	5 (−) 180	$u_1 = 0$
A_2	240	5 160	25 (−) 80	5 (+) -10	$u_2 = 10$
A_3	30	0 20	0 30	0 10	$u_3 = -15$
V		$v_1 = -5$	$v_2 = 15$	$v_3 = 5$	

Обсяг перерозподілу Δ : _____.

10. Оберіть правильні твердження щодо результатів застосування побудованого циклу переходу.

Споживачі / Постачальники		B_1	B_2	B_3	U
		130	110	210	
A_1	210	5 130	15 (−) 80	5 (+) -10	$u_1 = 0$
A_2	150	20 15	15 (+) 30	15 (−) 120	$u_2 = 0$
A_3	90	0 10	0 0	0 90	$u_3 = -15$
V		$v_1 = 5$	$v_2 = 15$	$v_3 = 15$	

а) Комірка c_{13} входить до базису.

б) Комірка c_{12} виходить із базису.

в) Обсяг поставки x_{13} у комірці c_{13} стає рівним 80.

г) Обсяг поставки x_{23} у комірці c_{23} стає рівним 40.

д) Обсяг поставки x_{22} у комірці c_{22} стає рівним 110.

е) Комірка c_{23} виходить із базису.

ж) Комірка c_{13} виходить із базису.

з) Комірка c_{12} входить до базису.

11. Оберіть правильні твердження щодо результатів застосування побудованого циклу переходу.

Споживачі / Постачальники		B_1	B_2	B_3	U
		130	110	170	
A_1	210	5 130	15 80	5 -10	$u_1 = 0$
A_2	110	20 15	15 30	15 80	$u_2 = 0$
A_3	90	0 10	0 0	0 90	$u_3 = -15$
V		$v_1 = 5$	$v_2 = 15$	$v_3 = 15$	

а) Комірка c_{13} входить до базису.

б) Комірка c_{12} або c_{23} виходить із базису.

в) Обсяг поставки x_{13} у комірці c_{13} стає рівним 80.

г) Обсяг поставки x_{12} у комірці c_{12} стає рівним 0.

д) Обсяг поставки x_{23} у комірці c_{23} стає рівним 0.

е) Обсяг поставки x_{22} у комірці c_{22} стає рівним 110.

ж) Комірка c_{13} виходить із базису.

з) Комірка c_{12} або c_{23} входить до базису.

ДОДАТОК В

ЗАВДАННЯ ПІДСУМКОВОГО ТЕСТУ

В.1 Основні властивості

1. Введіть значення параметрів запропонованої транспортної задачі.

Споживачі / Постачальники		B_1	B_2	B_3
		120	80	60
A_1	100	5	10	20
A_2	50	10	25	5
A_3	70	20	15	10
A_4	30	25	5	15

Кількість постачальників: _____;

Кількість споживачів: _____;

Загальний обсяг запасів постачальників: _____;

Загальний обсяг потреб споживачів: _____.

2. Визначте тип кожної транспортної задачі: відкритий чи закритий.

1. Транспортна задача відкритого типу
2. Транспортна задача закритого типу

1.

Споживачі / Постачальники		B_1	B_2	B_3
		30	20	50
A_1	20	5	20	25
A_2	80	15	15	5

2.

Споживачі / Постачальники		B_1	B_2	B_3	B_4
		20	30	80	70
A_1	110	5	20	15	0
A_2	90	15	5	10	0

3.

Споживачі / Постачальники		B_1	B_2	B_3
		30	70	60
A_1	80	15	5	20
A_2	80	5	25	10

4.

Споживачі / Постачальники		B_1	B_2
		80	30
A_1	10	5	20
A_2	70	5	10
A_3	20	15	20

5.

Споживачі / Постачальники		B_1	B_2
		40	60
A_1	50	5	20
A_2	70	20	25

3. Побудуйте правильні твердження стосовно приведення транспортної задачі до закритого типу.

Якщо загальний обсяг запасів постачальників дорівнює загальному обсягу потреб споживачів, тоді треба _____ (ввести фіктивного постачальника / ввести фіктивного споживача / ввести фіктивного постачальника та споживача / нічого не робити).

Якщо загальний обсяг потреб споживачів перевищує загальний обсяг запасів постачальників, тоді треба _____ (ввести фіктивного постачальника / ввести фіктивного споживача / ввести фіктивного постачальника та споживача / нічого не робити).

Якщо загальний обсяг запасів постачальників перевищує загальний об'єм потреб споживачів, тоді треба _____ (ввести фіктивного постачальника / ввести фіктивного споживача / ввести фіктивного постачальника та споживача / нічого не робити).

4. Обчисліть обсяг запасів/потреб фіктивних постачальників/споживачів у запропонованих транспортних задачах відкритого типу.

Споживачі / Постачальники		B_1	B_2	B_3
		30	70	?
A_1	50	5	15	0
A_2	60	20	25	0

Потреби b_3 фіктивного споживача B_3: _____.

Споживачі / Постачальники		B_1	B_2	B_3
		100	70	30
A_1	80	15	5	10
A_2	40	25	5	20
A_3	50	5	15	10
A_4	?	0	0	0

Запаси a_4 фіктивного постачальника A_4: _____.

В.2 План перевезень

1. Користуючись теоремою про ранг матриці коефіцієнтів системи обмежень транспортної задачі обчисліть необхідну кількість базисних клітин.

Транспортну задачу відкритого типу, де було 2 постачальників та 3 споживачів, привели до закритого типу. Кількість базисних клітин: _____.

2. Установіть послідовність розгляду клітин тарифів транспортної таблиці за методом північно-західного кута.

Примітка. Зверніть увагу: B_4 — фіктивний споживач.

Споживачі / Постачальники		B_1	B_2	B_3	B_4
		40	60	70	30
A_1	90	5	10	20	0
A_2	50	25	15	5	0
A_3	20	20	10	10	0
A_4	40	5	25	5	0

— c_{11};

— c_{12};

— c_{22};

— c_{23};

— c_{33};

— c_{43};

— c_{44}.

3. Установіть послідовність розгляду клітин тарифів транспортної таблиці за методом мінімальної вартості.

Примітка. Зверніть увагу: A_4 — фіктивний постачальник.

Споживачі / Постачальники		B_1	B_2	B_3
		50	70	50
A_1	40	5	20	15
A_2	60	10	15	35
A_3	60	30	25	20
A_4	10	0	0	0

— c_{11};

— c_{21};

— c_{22};

— c_{33};

— c_{32};

— c_{42}.

4. Обчисліть обсяг поставки x_{22} за тарифом у клітині c_{22} транспортної таблиці, а також оновлене значення запасу постачальника A_2 та потреб споживача B_2.

Споживачі / Постачальники		B_1	B_2
		70	100
A_1	90	25	10
A_2	80	15	5 ?

Обсяг поставки x_{22}: _____.

Оновлений обсяг запасів постачальника A_2: _____.

Оновлений обсяг потреб споживача B_2: _____.

5. Обчисліть обсяг поставки x_{23} за тарифом у клітині c_{23} транспортної таблиці, а також оновлене значення запасу постачальника A_2 та потреб споживача B_3.

Споживачі / Постачальники		B_1	B_2	B_3
		20	~~40~~ **0**	~~20~~ **10**
A_1	~~50~~ **0**	20	5 40	10 10
A_2	30	20	25	15 ?

Обсяг поставки x_{23}: _____.

Оновлений обсяг запасів постачальника A_2: _____.

Оновлений обсяг потреб споживача B_3: _____.

6. Обчисліть обсяг поставки x_{22} за тарифом у клітині c_{22} транспортної таблиці, а також оновлене значення запасу постачальника A_2 та потреб споживача B_2.

Споживачі / Постачальники		B_1	B_2
		~~60~~ **0**	40
A_1	20	10	25
A_2	~~80~~ **20**	5 60	20 ?

Обсяг поставки x_{22}: _____.

Оновлений обсяг запасів постачальника A_2: _____.

Оновлений обсяг потреб споживача B_2: _____.

7. Що треба зробити після виконаної поставки $x_{12} = 30$ у комірку c_{12}?

Примітка. Обраний варіант відповіді повинен відображати всі можливі дії.

Споживачі / Постачальники		B_1	B_2	B_3
		~~10~~ **0**	~~30~~ **0**	60
A_1	~~40~~ **0**	5 10	10 30	25
A_2	60	15	25	20

а) Видалити з розгляду постачальника A_1.

б) Видалити з розгляду споживача B_2.

в) Видалити з розгляду постачальника A_1 або споживача B_2.

г) Видалити з розгляду постачальника A_1 та споживача B_2.

8. Що треба зробити після виконаної поставки $x_{21} = 50$ у комірку c_{21}?

Примітка. Обраний варіант відповіді повинен відображати всі можливі варіанти.

Споживачі / Постачальники		B_1	B_2	B_3
		~~50~~ **0**	~~20~~ **0**	~~20~~ **0**
A_1	~~40~~ **0**	30	5 20	10 20
A_2	~~50~~ **0**	15 50	25	20 0

а) Видалити з розгляду постачальника A_2.

б) Видалити з розгляду споживача B_1.

в) Видалити з розгляду постачальника A_2 або споживача B_1.

г) Видалити з розгляду постачальника A_2 та споживача B_1.

9. Що треба зробити після виконаної поставки $x_{23} = 30$ у комірку c_{23}?

Примітка. Обраний варіант відповіді повинен відображати всі можливі варіанти.

Споживачі / Постачальники		B_1	B_2	B_3	B_4
		40	20	~~30~~ **0**	10
A_1	40	10	15	20	20
A_2	~~60~~ **30**	15	20	5 30	25

а) Видалити з розгляду постачальника A_2.

б) Видалити з розгляду споживача B_3.

в) Видалити з розгляду постачальника A_2 або споживача B_3.

г) Видалити з розгляду постачальника A_2 та споживача B_3.

10. Що треба зробити після виконаної поставки $x_{32} = 50$ у комірку c_{32}?

Примітка. Обраний варіант відповіді повинен відображати всі можливі варіанти.

Споживачі / Постачальники		B_1	B_2
		40	~~60~~ **10**
A_1	20	20	10
A_2	30	10	25
A_3	~~50~~ **0**	15	5 50

а) Видалити з розгляду постачальника A_3.

б) Видалити з розгляду споживача B_2.

в) Видалити з розгляду постачальника A_3 або споживача B_2.

г) Видалити з розгляду постачальника A_3 та споживача B_2.

11. Обчисліть загальну вартість перевезень за планом, що наведений у вигляді транспортної таблиці.

Споживачі / Постачальники		B_1	B_2
		30	70
A_1	10	5 [10]	20
A_2	50	10 [20]	10 [30]
A_3	40	25	5 [40]

Загальна вартість перевезень: _____.

В.3 Метод потенціалів

1. Оберіть правильні твердження щодо застосування методу потенціалів для розв’язання транспортної задачі.

а) Значення першого потенціалу обов’язково має дорівнювати 0.

б) Значення першого потенціалу є довільним.

в) Перший потенціал має визначатися для постачальника.

г) Перший потенціал має визначатися для споживача.

д) Починати визначення потенціалів можна з будь-якого учасника.

2. Обчисліть потенціал u_2 постачальника A_2 та v_2 споживача B_2.

Споживачі / Постачальники		B_1	B_2	B_3	U
		30	50	20	
A_1	20	5 [20]	20	25	
A_2	50	15 [10]	25 [20]	10 [20]	$u_2 = ?$
A_3	30	20	30 [30]	35	
V		$v_1 = 0$	$v_2 = ?$		

Потенціал u_2 постачальника A_2: _____.

Потенціал v_2 споживача B_2: _____.

3. Оберіть вільні комірки запропонованої транспортної таблиці.

Споживачі / Постачальники		B_1	B_2	B_3
		50	30	20
A_1	40	25 10	5 30	30
A_2	60	15 40	20	10 20

а) c_{11}.

б) c_{12}.

в) c_{13}.

г) c_{21}.

д) c_{22}.

е) c_{23}.

4. Обчисліть значення Δ_{13} для позначеної вільної комірки транспортної таблиці.

Споживачі / Постачальники		B_1	B_2	B_3	B_4	U
		140	100	190	60	
A_1	240	5 140	15 100	10 ?	0 0	$u_1 = 0$
A_2	250	15	25	5 190	0 60	$u_2 = 0$
V		$v_1 = 5$	$v_2 = 15$	$v_3 = 5$	$v_4 = 0$	

Значення Δ_{13} дорівнює _____.

5. Оберіть правильні твердження стосовно наведеного плану перевезень.

Споживачі / Постачальники		B_1	B_2	B_3	U
		220	120	20	
A_1	190	20 **170**	25 10	0 **20**	$u_1 = 0$
A_2	170	10 **50**	5 **120**	0 10	$u_2 = -10$
V		$v_1 = 20$	$v_2 = 15$	$v_3 = 0$	

а) Оптимальний.

б) Не оптимальний.

в) Оптимальний та єдиний.

г) Оптимальний та не єдиний.

6. Оберіть правильні твердження стосовно наведеного плану перевезень.

Споживачі / Постачальники		B_1	B_2	B_3	U
		160	170	350	
A_1	210	5 **160**	10 5	0 **50**	$u_1 = 0$
A_2	230	5 0	15 10	0 **230**	$u_2 = 0$
A_3	240	25 20	5 **170**	0 **70**	$u_3 = 0$
V		$v_1 = 5$	$v_2 = 5$	$v_3 = 0$	

а) Оптимальний.

б) Не оптимальний.

в) Оптимальний та єдиний.

г) Оптимальний та не єдиний.

7. Оберіть правильні твердження стосовно наведеного плану перевезень.

Постачальники \ Споживачі		B_1	B_2	B_3	U
		200	150	120	
A_1	170	25 · 50	10 · -10	0 · 120	$u_1 = 0$
A_2	150	20 · 5	10 · 150	0 · 10	$u_2 = -10$
A_3	150	25 · 150	20 · 0	0 · 0	$u_3 = 0$
V		$v_1 = 25$	$v_2 = 20$	$v_3 = 0$	

а) Оптимальний.

б) Не оптимальний.

в) Оптимальний та єдиний.

г) Оптимальний та не єдиний.

8. Оберіть правильно побудовані цикли переходу.

1.

Постачальники \ Споживачі		B_1	B_2	B_3	B_4	B_5	U
		150	160	200	210	20	
A_1	160	10 · -5	10 · -5	5 · 160	15 · 5	0 · 5	$u_1 = 0$
A_2	200	20 · 0	25 · 5	20 · 10	15 · 200	0 · 0	$u_2 = 5$
A_3	200	20 · 150	20 · 10	10 · 40	20 · 5	0 · 0	$u_3 = 5$
A_4	180	20 · 0	20 · 150	15 · 5	15 · 10	0 · 20	$u_4 = 5$
V		$v_1 = 15$	$v_2 = 15$	$v_3 = 5$	$v_4 = 10$	$v_5 = -5$	

2.

Споживачі / Постачальники		B_1	B_2	B_3	B_4	B_5	U
		150	160	200	210	20	
A_1	160	10 -5	10 -5	5 160	15 5	0 5	$u_1 = 0$
A_2	200	20 0	25 5	20 10	15 200	0 0	$u_2 = 5$
A_3	200	20 150	20 10	10 40	20 5	0 0	$u_3 = 5$
A_4	180	20 0	20 150	15 5	15 10	0 20	$u_4 = 5$
V		$v_1 = 15$	$v_2 = 15$	$v_3 = 5$	$v_4 = 10$	$v_5 = -5$	

3.

Споживачі / Постачальники		B_1	B_2	B_3	B_4	B_5	U
		150	160	200	210	20	
A_1	160	10 -5	10 -5	5 160	15 5	0 5	$u_1 = 0$
A_2	200	20 0	25 5	20 10	15 200	0 0	$u_2 = 5$
A_3	200	20 150	20 10	10 40	20 5	0 0	$u_3 = 5$
A_4	180	20 0	20 150	15 5	15 10	0 20	$u_4 = 5$
V		$v_1 = 15$	$v_2 = 15$	$v_3 = 5$	$v_4 = 10$	$v_5 = -5$	

4.

Споживачі / Постачальники		B_1	B_2	B_3	B_4	B_5	U
		150	160	200	210	20	
A_1	160	10 -5	10 -5	5 160	15 5	0 5	$u_1 = 0$
A_2	200	20 0	25 5	20 10	15 200	0 0	$u_2 = 5$
A_3	200	20 150	20 10	10 40	20 5	0 0	$u_3 = 5$
A_4	180	20 0	20 150	15 5	15 10	0 20	$u_4 = 5$
V		$v_1 = 15$	$v_2 = 15$	$v_3 = 5$	$v_4 = 10$	$v_5 = -5$	

5.

Споживачі / Постачальники		B_1	B_2	B_3	B_4	B_5	U
		150	160	200	210	20	
A_1	160	10 -5	10 -5	5 160	15 5	0 5	$u_1 = 0$
A_2	200	20 0	25 5	20 10	15 200	0 0	$u_2 = 5$
A_3	200	20 150	20 10	10 40	20 5	0 0	$u_3 = 5$
A_4	180	20 0	20 150	15 5	15 10	0 20	$u_4 = 5$
V		$v_1 = 15$	$v_2 = 15$	$v_3 = 5$	$v_4 = 10$	$v_5 = -5$	

6.

Постачальники \ Споживачі		B_1	B_2	B_3	B_4	B_5	U
		150	160	200	210	20	
A_1	160	10 -5	10 -5	5 160	15 5	0 5	$u_1 = 0$
A_2	200	20 0	25 5	20 10	15 200	0 0	$u_2 = 5$
A_3	200	20 150	20 10	10 40	20 5	0 0	$u_3 = 5$
A_4	180	20 0	20 150	15 5	15 10	0 20	$u_4 = 5$
V		$v_1 = 15$	$v_2 = 15$	$v_3 = 5$	$v_4 = 10$	$v_5 = -5$	

9. Обчисліть обсяг перерозподілу Δ для побудованого циклу переходу.

Постачальники \ Споживачі		B_1	B_2	B_3	U
		210	190	150	
A_1	200	20 30	5 20	0 150	$u_1 = 0$
A_2	170	10 -10	5 170	0 0	$u_2 = 0$
A_3	180	10 180	15 20	0 10	$u_3 = -10$
V		$v_1 = 20$	$v_2 = 5$	$v_3 = 0$	

Обсяг перерозподілу Δ: _____.

10. Оберіть правильні твердження щодо результатів застосування побудованого циклу переходу.

Споживачі / Постачальники		B_1	B_2	B_3	U
		120	120	390	
A_1	210	25	20	0 -5	$u_1 = 0$
		120	90		
A_2	240	25 5	15	0	$u_2 = -5$
			30	210	
A_3	180	25 5	15 0	0	$u_3 = -5$
				180	
V		$v_1 = 25$	$v_2 = 20$	$v_3 = 5$	

а) Комірка c_{13} входить до базису.

б) Комірка c_{12} виходить із базису.

в) Обсяг поставки x_{13} у комірці c_{13} стає рівним 90.

г) Обсяг поставки x_{23} у комірці c_{23} стає рівним 120.

д) Обсяг поставки x_{22} у комірці c_{22} стає рівним 120.

е) Комірка c_{23} виходить із базису.

ж) Комірка c_{13} виходить із базису.

з) Комірка c_{12} входить до базису.

11. Оберіть правильні твердження щодо результатів застосування побудованого циклу переходу.

Споживачі / Постачальники		B_1	B_2	B_3	U
		190	210	280	
A_1	250	10 ⊖ 190	15 ⊕ 60	0 10	$u_1 = 0$
A_2	230	25 5	25 ⊖ 150	0 ⊕ 80	$u_2 = 10$
A_3	200	10 ⊕ -10	25 0	0 ⊖ 200	$u_3 = 10$
V		$v_1 = 10$	$v_2 = 15$	$v_3 = -10$	

а) Комірка c_{31} входить до базису.

б) Комірка c_{22} виходить із базису.

в) Обсяг поставки x_{11} у комірці c_{11} стає рівним 40.

г) Обсяг поставки x_{33} у комірці c_{33} стає рівним 50.

д) Обсяг поставки x_{12} у комірці c_{12} стає рівним 210.

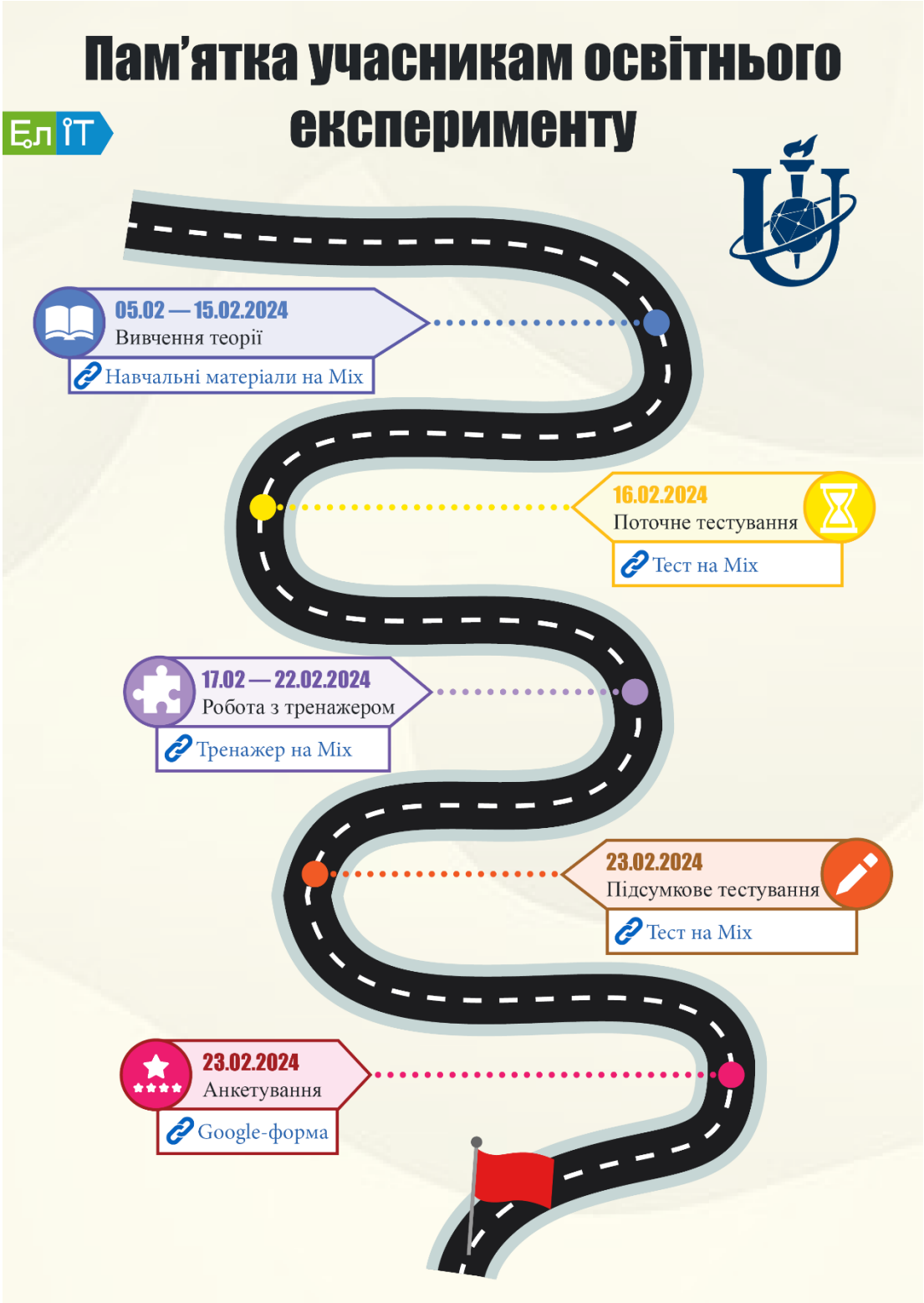
е) Обсяг поставки x_{23} у комірці c_{23} стає рівним 230.

ж) Комірка c_{11} виходить із базису.

з) Комірка c_{33} виходить із базису.

ДОДАТОК Г

ПАМ’ЯТКА УЧАСНИКАМ ОСВІТНЬОГО ЕКСПЕРИМЕНТУ



ДОДАТОК Д

ІНСТРУКЦІЯ ДО ТРЕНАЖЕРА ДЛЯ ЕКСПЕРИМЕНТАЛЬНОЇ ГРУПИ

Інструкція до тренажера “Транспортні задачі“

Основні відомості

Тренажер дозволяє послідовно пройти всі кроки розв’язання згенерованої транспортної задачі: від приведення до закритого типу до застосування методу потенціалів.

Завдання тренажера мають різну вагу. До того ж деякі організовані в цикли, тому оновлення прогресу виконання не є рівномірним.

Графічний компонент тренажера у вигляді цифрового годинника демонструє поточну тривалість виконання. Він не обмежує Вас у часі.

Для зарахування проходження етапу «Робота з тренажером» Вам необхідно мати мінімум 1 позитивну спробу (60% або більше). Режим виконання неважливий.

На платформі Mix фіксується найвищий результат виконання, тому не бійтеся тренуватися більше. Таким чином Вам вдасться зустрітися з усіма можливими тонкощами.

Режими тренажера

У режимах «Тренування» та «Своя задача» бали за виправлення не знімаються.

Режим «Своя задача» надає можливість введення конкретної задачі або перегенерації запропонованої на початку.

У режимі «Екзамен» використовується сувора система оцінювання: якщо припуститися однієї або більше помилок під час виконання окремого завдання або їхнього циклу — максимальна оцінка за нього стає 50% від початкової (не накопичується). Контекстні підказки після помилок не надаються. Рекомендуємо спробувати, коли будете впевнені у своїх знаннях.

1

Умовні позначення

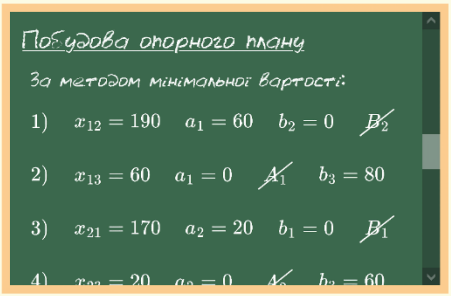
Формульні позначення відповідають тим, що використовуються в навчальних матеріалах дисципліни:

Формула	Значення
i	Номер постачальника
j	Номер споживача
m	Кількість постачальників
n	Кількість споживачів
A_i	i-тий постачальник
B_j	j-тий споживач
a_i	Запаси постачальника A_i
b_j	Потреби споживача B_j
u_i	Потенціал постачальника A_i
v_j	Потенціал споживача B_j
c_{ij}	Вартість перевезення 1 одиниці вантажу між постачальником A_i та споживачем B_j
x_{ij}	Обсяг поставки від постачальника A_i до споживача B_j
X^k	План перевезень k
X^*	Оптимальний план перевезень
N	Кількість базисних комірок
E_{ij}	Характеристика вільної клітини між постачальником A_i та споживачем B_j
f	Цільова функція
Δ	Обсяг перерозподілу за циклом переходу

Зверніть увагу на особливості розташування значень **атрибутів** комірок тарифів та **колір заливки**:

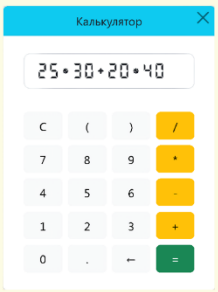
Можливості тренажера

Слідкуйте за формалізованим описом процесу розв'язання задачі на віртуальній дошці.



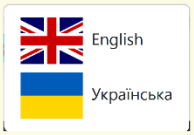
Віртуальний електронний годинник показує поточний час виконання, щоб Ви мали змогу оцінити свою продуктивність. Він не накладає обмежень на час виконання.

Користуйтеся вбудованим калькулятором. Наявна підтримка складних виразів дозволить спростити обчислення.



Бажаєте повторити процес розв'язання без втрати згенерованої умови задачі? Скористайтеся кнопкою перезапуску тренажера (доступна в режимах «Тренування» та «Своя задача»).

Тренажер підтримує декілька локалізацій: англійську та українську. Між ними можна миттєво перемикатися в будь-який момент.



Якщо вирішили завчасно завершити роботу — є кнопка «Завершити тренажер».

Ергономіка

- **Фокусування поля** введення числової відповіді відбувається автоматично після появи завдання. Для **скидання фокуса** — «Esc».
- Можна використовувати клавішу «Enter» для **підтвердження введення** числової відповіді на завдання.
- У калькуляторі працює **автодоповнення** (вставляються дужки, додається/прибирається провідний нуль). Доступне керування за допомогою гарячих клавіш:
  - цифрові клавіші від 0 до 9 — для введення чисел;
  - точка — роздільник для десяткових чисел;
  - «Esc» — очищення вводу/результату;
  - «Backspace» або «Delete» — видалення останнього символу;
  - дужки та математичні операції вводяться за допомогою відповідних клавіш зі сполученням «Shift»;
  - натискання «=» призводить до обчислення результату.
- По завершенню побудови циклу переходу **замкнення** відбувається автоматично.

Зичимо Вам успіху!

ДОДАТОК Е

ІНСТРУКЦІЯ ДО ТРЕНАЖЕРА ДЛЯ КОНТРОЛЬНОЇ ГРУПИ

Інструкція до тренажера “Транспортні задачі“

Основні відомості

Тренажер має вигляд динамічного тесту: під час кожного запуску відбувається генерація завдань, що являють собою окремі ситуації з процесу розв’язання не пов’язаних між собою транспортних задач.

Для зарахування проходження етапу «Робота з тренажером» Вам необхідно мати мінімум 1 позитивну спробу (60% або більше).

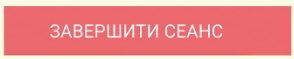
На платформі Mix фіксується найвищий результат виконання, тому не бійтеся тренуватися більше.

Можливості тренажера

Користуйтеся вбудованим калькулятором. Наявна підтримка складних виразів дозволить спростити обчислення.

Тренажер підтримує декілька локалізацій: українську та англійську. Оберіть мову перед початком роботи.

Якщо вирішили завчасно завершити роботу — є кнопка «Завершити сеанс».



1

Умовні позначення

Формульні позначення відповідають тим, що використовуються в навчальних матеріалах дисципліни:

Формула	Значення
i	Номер постачальника
j	Номер споживача
m	Кількість постачальників
n	Кількість споживачів
A_i	i-тий постачальник
B_j	j-тий споживач
a_i	Запаси постачальника A_i
b_j	Потреби споживача B_j
u_i	Потенціал постачальника A_i
v_j	Потенціал споживача B_j
c_{ij}	Вартість перевезення 1 одиниці вантажу між постачальником A_i та споживачем B_j
x_{ij}	Обсяг поставки від постачальника A_i до споживача B_j
X^k	План перевезень k
X^*	Оптимальний план перевезень
N	Кількість базисних комірок
E_{ij}	Характеристика вільної клітини між постачальником A_i та споживачем B_j
f	Цільова функція
Δ	Обсяг перерозподілу за циклом переходу

Зверніть увагу на особливості розташування значень **атрибутів** комірок тарифів та **колір заливки**:

Зичимо Вам успіху!

ДОДАТОК Ж

АНКЕТА УЧАСНИКА ОСВІТНЬОГО ЕКСПЕРИМЕНТУ

Шановний студенте! Ми вдячні Вам за участь у нашому експерименті «Ефективність застосування в освітньому процесі віртуальних навчальних тренажерів на основі автоматних моделей», що проводиться на базі кафедри комп'ютерних наук Сумського державного університету. Інформація, що збирається, використовуватиметься в знеособленому вигляді для перевірки гіпотез щодо наявності статистично значущого позитивного впливу на різні показники, зокрема рівень успішності, часові витрати на виконання завдань, мотивацію, повноту моделювання задач та ін. Отримані Вами навчальні досягнення та враження стануть важливим внеском у розвиток системи вищої освіти в Україні.

Прізвище, ім'я, по-батькові

_____

Як Ви оцінюєте вплив роботи з тренажером на Ваш рівень мотивації до навчання?

а) Виключно негативний
б) Переважно негативний
в) Нейтральний
г) Переважно позитивний
д) Виключно позитивний

Чи згодні Ви з тим, що запропонований тренажер повноцінно моделює процес розв'язання транспортних задач?

а) Повністю погоджуюся
б) Частково погоджуюся

в) Не знаю

г) Частково не погоджуюся

д) Повністю не погоджуюся

Які проблеми у Вас виникли під час роботи з тренажером?

_____

_____

Які у Вас залишилися побажання щодо подальшого розроблення навчального додатку?

_____

_____

ДОДАТОК З

ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАВДАНЬ ТРЕНАЖЕРА НА ОСНОВІ REACT

Об’єктна модель потоку завдань (task-flow.ts):

```
import { Flowable } from "../models/interfaces/Flowable";
import { LinearTaskFlow } from "../models/flows/LinearTaskFlow";
import { SketchDescriptor } from
"../models/descriptors/sketches/SketchDescriptor";
import { SwitchTaskFlow } from "../models/flows/SwitchTaskFlow";
import { WhileLoopTaskFlow } from "../models/flows/WhileLoopTaskFlow";
import { TariffCellDescriptor } from
"../models/descriptors/cells/TariffCellDescriptor";
import { FlowableAction } from "../models/flows/FlowableAction";
import { MouseSelectionAnswer } from "../models/answers/MouseSelectionAnswer";
import { _formula, _i, _j, _X } from "./configuration";
import { HeadingSketch } from "../models/sketches/HeadingSketch";
import { FormulaSketchDisplayMode } from
"../models/enums/FormulaSketchDisplayMode";
import { SeparatorSketch } from "../models/sketches/SeparatorSketch";
import { FormulaSketch, FormulaSketchProps } from
"../models/sketches/FormulaSketch";
import { ParagraphSketch } from "../models/sketches/ParagraphSketch";
import { ParagraphSketchTextAlign } from
"../models/enums/ParagraphSketchTextAlign";
import { ParticipantType } from "../models/enums/ParticipantType";
import { CellDescriptor } from "../models/descriptors/cells/CellDescriptor";
import { TransitionCycleAnswer } from "../models/answers/TransitionCycleAnswer";
import { forEachTariff } from "../utils/task-flow";
import { SpaceSketch } from "../models/sketches/SpaceSketch";
import { supplierCountTask } from "./tasks/supplier-count";
import { consumerCountTask } from "./tasks/consumer-count";
import { totalProductionTask } from "./tasks/total-production";
import { totalConsumptionTask } from "./tasks/total-consumption";
import { totalProductionAndTotalConsumptionComparison } from "./tasks/total-
production-and-total-consumption-comparison";
import { problemTypeTask } from "./tasks/problem-type";
import { requiredProblemTransformationsTask } from "./tasks/required-problem-
transformations";
import { newParticipantNumberTask } from "./tasks/new-participant-number";
import { newParticipantProductCountTask } from "./tasks/new-participant-product-
count";
import { newParticipantTariffsTask } from "./tasks/new-participant-tariffs";
import { basicPlanMethod } from "./tasks/basic-plan-method";
import { cellByNorthwestCornerMethodTask } from "./tasks/cell-by-northwest-
corner-method";
import { cellByLeastCostMethod } from "./tasks/cell-by-least-cost-method";
import { cellByCustomMethod } from "./tasks/cell-by-custom-method";
import { shipmentTask } from "./tasks/shipment";
import { supplierAndConsumerProductCountRemainingTask } from "./tasks/supplier-
and-consumer-product-count-remaining";
import { participantEliminationTask } from "./tasks/participant-elimination";
```

```
import { afterSupplierEliminationTask } from "./tasks/after-supplier-
elimination";
import { afterConsumerEliminationTask } from "./tasks/after-consumer-
elimination";
import { supplierProductCountRemainingTask } from "./tasks/supplier-product-
count-remaining";
import { consumerProductCountRemainingTask } from "./tasks/consumer-product-
count-remaining";
import { requiredBasicCellCountTask } from "./tasks/required-basic-cell-count";
import { basicCellCountTask } from "./tasks/basic-cell-count";
import { rankTheoremTask } from "./tasks/rank-theorem";
import { transportationCostTask } from "./tasks/transportation-cost";
import { smallPlanTask } from "./tasks/small-plan";
import { potentialTask } from "./tasks/potential";
import { potentialValueTask } from "./tasks/potential-value";
import { nonBasicCellTask } from "./tasks/non-basic-cell";
import { characteristicValueTask } from "./tasks/characteristic-value.ts";
import { planTask } from "./tasks/plan";
import { transitionCycleTask } from "./tasks/transition-cycle";
import { transitionValueTask } from "./tasks/transition-value";
import { basicCellEliminationTask } from "./tasks/basic-cell-elimination";

export const buffer: {
    totalProduction?: number;
    totalConsumption?: number;
    newSupplierRequired?: boolean;
    newConsumerRequired?: boolean;
    openTransportationProblem?: boolean;
    totalProductionFormulaDescriptor?: SketchDescriptor;
    totalConsumptionFormulaDescriptor?: SketchDescriptor;
    totalProductionAndTotalConsumptionComparisonFormulaDescriptor?:
SketchDescriptor;
    newParticipantNumberSymbol?: string;
    newParticipantNumber?: number;
    newParticipantProductCount?: number;
    basicPlanMethod?: string;
    tariffCell?: TariffCellDescriptor;
    shipment?: number;
    eliminatedSupplierCount?: number;
    eliminatedConsumerCount?: number;
    eliminatedParticipantType?: ParticipantType;
    notEliminatedParticipantIndex?: number;
    requiredBasicCellCount?: number;
    basicCellCount?: number;
    planIndex?: number;
    optimalPlan?: boolean;
    uniquePlan?: boolean;
    potentialCount?: number;
    pairedPotential?: number;
    pairedPotentialCell?: CellDescriptor;
    potentialCell?: CellDescriptor;
    nonBasicCellCount?: number;
    minCharacteristic?: number;
    basicCellElimination?: boolean;

    requiredBasicCellCountFormulaDescriptor?: SketchDescriptor;
    basicCellCountFormulaDescriptor?: SketchDescriptor;
} = {};

export const trainerTaskFlow: Flowable = new LinearTaskFlow([
    //-------------------------------------------------------------------------
----------------------------------------
    // Етап 1: основні властивості, приведення задачі до закритого типу
```

```
    //--------------------------------------------------------------------------
----------------------------------------
    // Ініціалізація потоку завдань: виведення основних написів на дошці (назва
процесу, основні властивості)
    new FlowableAction((taskStore) => {
        taskStore.addSketches(
            new HeadingSketch({
                id: "title",
                level: 1,
            }),
            new HeadingSketch({
                id: "mainPropertiesHeading",
                level: 2,
            }),
        );
    }),

    supplierCountTask,
    consumerCountTask,

    // Додавання невидимого роздільника, щоб наступні формули на дошці
з'являлися з нового рядка
    new FlowableAction((taskStore) => {
        taskStore.addSketch(
            new SeparatorSketch({
                visible: false,
            }),
        );
    }),

    totalProductionTask,
    totalConsumptionTask,
    totalProductionAndTotalConsumptionComparison,
    problemTypeTask,

    // Завершення блоку основних властивостей задачі
    new FlowableAction((taskStore) => {
        taskStore.addSketch(new SeparatorSketch());
    }),

    // Підтримка задач відкритого типу: за необхідності відбудеться введення
фіктивного постачальника або споживача
    new SwitchTaskFlow(
        [
            {
                condition: (taskStore) => buffer.openTransportationProblem!,
                flowable: new LinearTaskFlow([
                    // Початок блоку та обчислення допоміжних даних про
фіктивного учасника перевезень:
                    // - символ
                    // - номер
                    // - кількість товару (запаси або споживання)
                    new FlowableAction((taskStore) => {
                        if (buffer.newSupplierRequired) {
                            buffer.newParticipantNumberSymbol = _i;
                            buffer.newParticipantNumber =
taskStore.suppliers.length + 1;
                            buffer.newParticipantProductCount =
buffer.totalConsumption! - buffer.totalProduction!;
                        } else {
                            buffer.newParticipantNumberSymbol = _j;
                            buffer.newParticipantNumber =
taskStore.consumers.length + 1;
```

```
                            buffer.newParticipantProductCount =
buffer.totalProduction! - buffer.totalConsumption!;
                        }

                        taskStore.addSketch(
                            new HeadingSketch({
                                id: "problemTransformationHeading",
                                level: 2,
                            }),
                        );
                    }),

                    requiredProblemTransformationsTask,
                    newParticipantNumberTask,
                    newParticipantProductCountTask,
                    newParticipantTariffsTask,

                    // Завершення блоку введення фіктивного учасника перевезень
                    new FlowableAction((taskStore) => {
                        taskStore.addSketch(new SeparatorSketch());
                    }),
                ]),
            },
        ],
        5,
    ),

    //---------------------------------------------------------------------------
-----------------------------------
    // Етап 2: побудова опорного плану
    //---------------------------------------------------------------------------
-----------------------------------
    // Ініціалізація блоку, виведення його назви
    new FlowableAction((taskStore) => {
        buffer.eliminatedSupplierCount = 0;
        buffer.eliminatedConsumerCount = 0;
        buffer.basicCellCount = 0;
        taskStore.addSketch(
            new HeadingSketch({
                id: "basicPlanHeading",
                level: 2,
            }),
        );
    }),
    basicPlanMethod,

    // Побудова опорного плану
    new WhileLoopTaskFlow(
        {
            // Якщо вибули не всі учасники перевезень - формуємо завдання
            condition: (taskStore) =>
                buffer.eliminatedSupplierCount! < taskStore.suppliers.length ||
                buffer.eliminatedConsumerCount! < taskStore.consumers.length,
            flowable: new LinearTaskFlow([
                // Нумерація ітерацій
                new FlowableAction((taskStore) => {
                    taskStore.addSketch(
                        new FormulaSketch<FormulaSketchProps>({
                            formula: `${buffer.basicCellCount! + 1})`,
                            displayMode: FormulaSketchDisplayMode.PARTIAL_WIDTH,
                        }),
                    );
                }),
```

```
                // Завдання на виділення комірки тарифу за обраним методом
                new SwitchTaskFlow([
                    {
                        condition: (taskStore) => buffer.basicPlanMethod ===
"leastCost",
                        flowable: cellByLeastCostMethod,
                    },
                    {
                        condition: (taskStore) => buffer.basicPlanMethod ===
"northwestCorner",
                        flowable: cellByNorthwestCornerMethodTask,
                    },
                    {
                        condition: (taskStore) => buffer.basicPlanMethod ===
"custom",
                        flowable: cellByCustomMethod,
                    },
                ]),

                // Збереження виділення користувача
                new FlowableAction((taskStore) => {
                    buffer.tariffCell = (taskStore.answer as
MouseSelectionAnswer).selection as TariffCellDescriptor;
                    taskStore.select(buffer.tariffCell);
                }),

                shipmentTask,

                // Обрання алгоритму опрацювання залишків запасів постачальників
та потреб користувачів (якщо рівні -
                // достатньо лише 1 завдання)
                new SwitchTaskFlow([
                    {
                        // Запаси відповідають потребам
                        condition: (taskStore) =>

taskStore.suppliers[buffer.tariffCell!.supplierIndex].remainingProductCount ===

taskStore.consumers[buffer.tariffCell!.consumerIndex].remainingProductCount,
                        flowable: new LinearTaskFlow([
                            supplierAndConsumerProductCountRemainingTask,

                            // Перевірка на завершення побудови
                            new SwitchTaskFlow([
                                {
                                    // Комірка є останньою - можна вважати
постачальника та споживача вибулими
                                    condition: (taskStore) =>
                                        buffer.eliminatedSupplierCount! + 1 ===
taskStore.suppliers.length &&
                                        buffer.eliminatedConsumerCount! + 1 ===
taskStore.consumers.length,
                                    flowable: new FlowableAction((taskStore) =>
{
                                        const updatedSupplier =

taskStore.suppliers[buffer.tariffCell!.supplierIndex].clone();
                                        const updatedConsumer =

taskStore.consumers[buffer.tariffCell!.consumerIndex].clone();

                                        updatedSupplier.eliminated = true;
```

```
                                        buffer.eliminatedSupplierCount!++;

                                        updatedConsumer.eliminated = true;
                                        buffer.eliminatedConsumerCount!++;


taskStore.updateSupplier(buffer.tariffCell!.supplierIndex, updatedSupplier);

taskStore.updateConsumer(buffer.tariffCell!.consumerIndex, updatedConsumer);
                                        taskStore.addSketches(
                                            new
FormulaSketch<FormulaSketchProps>({
                                                formula:
`\\cancel{${updatedSupplier.labelFormula}}`,
                                                displayMode:
FormulaSketchDisplayMode.PARTIAL_WIDTH,
                                            }),
                                            new
FormulaSketch<FormulaSketchProps>({
                                                formula:
`\\cancel{${updatedConsumer.labelFormula}}`,
                                                displayMode:
FormulaSketchDisplayMode.PARTIAL_WIDTH,
                                            }),
                                        );
                                    }),
                                },
                                {
                                    // Комірка не остання - треба усунути або
постачальника, або споживача, щоб уникнути
                                    // втрати базисної комірки
                                    condition: (taskStore) => true,
                                    flowable: new LinearTaskFlow([
                                        participantEliminationTask,

                                        // Виділення комірки всередині елемента
таблиці, що залишився (рядок або стовпчик)
                                        // після вибування одного з учасників
перевезень (споживача або постачальника
                                        // відповідно)
                                        new SwitchTaskFlow([
                                            {
                                                condition: (taskStore) =>

buffer.eliminatedParticipantType === ParticipantType.SUPPLIER,
                                                flowable:
afterSupplierEliminationTask,
                                            },
                                            {
                                                condition: (taskStore) => true,
                                                flowable:
afterConsumerEliminationTask,
                                            },
                                        ]),
                                    ]),
                                },
                            ]),
                        ]),
                    },
                    {
                        // Запаси не відповідають потребам
                        condition: (taskStore) => true,
                        flowable: new LinearTaskFlow([
```

```
                            supplierProductCountRemainingTask,
                            consumerProductCountRemainingTask,
                        ]),
                    },
                ]),

                // Завершення ітерації: нова базисна комірка, додається
невидимий роздільник для подальшого виведення формул
                // на дощі з нового рядка
                new FlowableAction((taskStore) => {
                    buffer.basicCellCount!++;
                    taskStore.addSketch(
                        new SeparatorSketch({
                            visible: false,
                        }),
                    );
                }),
            ]),
        },
        20,
    ),

    //-------------------------------------------------------------------------
-----------------------------------------
    // Етап 3: застосування методу потенціалів
    //-------------------------------------------------------------------------
-----------------------------------------
    // Обчислення додаткових даних про побудований опорний план перевезень:
    // - кількість базисних клітин (за теоремою про ранг)
    // - ініціалізація значень властивостей плану:
    //   - індекс
    //   - кількість визначених потенціалів
    //   - оптимальність
    //   - унікальність
    new FlowableAction((taskStore) => {
        buffer.requiredBasicCellCount = taskStore.suppliers.length +
taskStore.consumers.length - 1;
        buffer.planIndex = 0;
        buffer.potentialCount = 0;
        buffer.optimalPlan = false;
        buffer.uniquePlan = true;

        taskStore.addSketch(
            new HeadingSketch({
                id: "rankTheoremHeading",
                level: 2,
            }),
        );
    }),

    requiredBasicCellCountTask,
    basicCellCountTask,
    rankTheoremTask,

    // Враховуємо транспортні задачі мінімальних розмірів: коли кількість
постачальників або споживачів дорівнює 1. У
    // такому випадку оптимізація неможлива - всі комірки входитимуть до базису.
    new SwitchTaskFlow(
        [
            {
                // Задача мінімального розміру, оптимізація розв'язку якої
неможлива
```

```
                condition: (taskStore) => taskStore.suppliers.length === 1 ||
taskStore.consumers.length === 1,
                flowable: new LinearTaskFlow([smallPlanTask,
transportationCostTask.clone()]),
            },
            {
                // Звичайна задача
                condition: (taskStore) => true,
                flowable: new LinearTaskFlow([
                    // Відображення комірок для застосування методу потенціалів
                    new FlowableAction((taskStore) => {
                        taskStore.setShowPotentials(true);
                        taskStore.addSketches(
                            new SeparatorSketch(),
                            new HeadingSketch({
                                id: "potentialMethod",
                                level: 2,
                            }),
                        );
                    }),

                    // Оптимізація опорного плану перевезень:
                    // - визначення потенціалів для постачальників та споживачів
                    // - обчислення оцінок вільних комірок
                    // - перевірка плану перевезень на оптимальність
                    // - побудова циклу переходу, якщо розв'язок не оптимальний
                    new WhileLoopTaskFlow({
                        condition: (taskStore) => !buffer.optimalPlan,
                        flowable: new LinearTaskFlow([
                            // Скидання потенціалів, якщо ті були визначені
раніше
                            new FlowableAction((taskStore) => {
                                const placeholderToValue = new Map<string,
string>();
                                placeholderToValue.set("planFormula",
_formula(`${_X}^{(${buffer.planIndex})}`));
                                taskStore.addSketches(
                                    new ParagraphSketch({
                                        id: "plan",
                                        align: ParagraphSketchTextAlign.CENTER,
                                        placeholderToValue,
                                    }),
                                    new ParagraphSketch({
                                        id: "potentials",
                                    }),
                                );

                                // Якщо потенціали ще не визначені (план є
першим) - пропускаємо обробку
                                if (buffer.potentialCount === 0) return;

                                // Для учасників перевезень потенціали раніше
були визначені
                                const updatedSuppliers =
taskStore.suppliers.map((supplier) => {
                                    const updatedSupplier = supplier.clone();
                                    updatedSupplier.potential = undefined;
                                    return updatedSupplier;
                                });
                                const updatedConsumers =
taskStore.consumers.map((consumer) => {
                                    const updatedConsumer = consumer.clone();
                                    updatedConsumer.potential = undefined;
```

```
                                    return updatedConsumer;
                                });
                                taskStore.setSuppliers(updatedSuppliers);
                                taskStore.setConsumers(updatedConsumers);
                                buffer.potentialCount = 0;

                                // Скидання characteristic для тарифів, у яких
вона була встановлена
                                forEachTariff(taskStore, (tariff, supplierIndex,
consumerIndex) => {
                                    if (tariff.characteristic !== undefined) {
                                        const updatedTariff = tariff.clone();
                                        updatedTariff.characteristic =
undefined;
                                        taskStore.updateTariff(supplierIndex,
consumerIndex, updatedTariff);
                                    }
                                });
                            }),

                            // Визначення потенціалів для постачальників та
споживачів:
                            // - вибір комірки потенціалу
                            // - введення значення потенціалу
                            new WhileLoopTaskFlow({
                                condition: (taskStore) =>
                                    buffer.potentialCount! <
taskStore.suppliers.length + taskStore.consumers.length,
                                flowable: new LinearTaskFlow([potentialTask,
potentialValueTask]),
                            }),

                            // Ініціалізація роботи із вільними комірками
                            new FlowableAction((taskStore) => {
                                buffer.nonBasicCellCount = 0;
                                buffer.minCharacteristic = undefined;

                                taskStore.addSketches(
                                    new SpaceSketch({
                                        size: 0.5,
                                    }),
                                    new ParagraphSketch({
                                        id: "characteristics",
                                    }),
                                );
                            }),

                            // Обчислення оцінок вільних комірок
                            new WhileLoopTaskFlow({
                                condition: (taskStore) =>
                                    buffer.nonBasicCellCount! <
                                    taskStore.tariffs
                                        .flat()
                                        .map((tariff) => (tariff.basic ? 0 : 1)
as number)
                                        .reduce((accumulator, current) =>
accumulator + current, 0),
                                flowable: new LinearTaskFlow([nonBasicCellTask,
characteristicValueTask]),
                            }),

                            new FlowableAction((taskStore) => {
                                taskStore.addSketch(
```

```
                                    new SpaceSketch({
                                        size: 0.5,
                                    }),
                                );
                            }),

                            planTask,
                            transportationCostTask,

                            // Перевірка доцільності застосування циклу переходу
                            new SwitchTaskFlow([
                                {
                                    // План не оптимальний - треба будувати
                                    condition: (taskStore) =>
!buffer.optimalPlan,
                                    flowable: new LinearTaskFlow([
                                        transitionCycleTask,
                                        transitionValueTask,

                                        // Якщо баланс порушено - треба
визначити, яка саме комірка з productCount = 0
                                        // повинна покинути базис. Обрана
комірка повинна належати раніше побудованому циклу
                                        // переходу - бути його вузлом зі знаком
"-".
                                        new SwitchTaskFlow([
                                            {
                                                // Баланс порушено - існує
потреба обрати лише одну комірку, яку треба
                                                // прибрати з базису
                                                condition: (taskStore) =>
buffer.basicCellElimination!,
                                                flowable:
basicCellEliminationTask,
                                            },
                                        ]),

                                        // Знищення раніше побудованої циклу
переходу
                                        new FlowableAction((taskStore) => {
                                            taskStore.setCycle(new
TransitionCycleAnswer());
                                            taskStore.addSketch(new
SpaceSketch());
                                        }),
                                    ]),
                                },
                            ]),
                        ]),
                    }),
                ]),
            },
        ],
        30,
    ),
]);
```

ДОДАТОК И

ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАВДАНЬ ТРЕНАЖЕРА НА ОСНОВІ JTRAINER

Спільний код кроків для роботи з транспортними задачами (trainer.custom.js):

```
window.jTrainerExt = {};

jTrainerExt.Direction = {
    RIGHT: 0,
    TOP: 1,
    LEFT: 2,
    BOTTOM: 3,
};

jTrainerExt.SubcellAlignment = {
    RIGHT: 0,
    TOP: 1,
    LEFT: 2,
    BOTTOM: 3,
    CENTER: 4,
};

jTrainerExt.CycleElement = {
    NONE: 0,
    NODE: 1,
    HORIZONTAL_LINK: 2,
    VERTICAL_LINK: 3,
    HORIZONTAL_AND_VERTICAL_LINKS: 4,
};

jTrainerExt.BasicPlanConstructionMethod = {
    LEAST_COST: 0,
    NORTHWEST_CORNER: 1,
}

jTrainerExt.TransportationProblemPreset = {
    ANY: [
        2, 4, 150, 250, 10,
        3, 5, 100, 230, 10,
        5, 25, 5
    ],
    OPEN: [
        2, 4, 150, 250, 10,
        3, 5, 100, 230, 10,
        5, 25, 5, true
    ],
    CLOSED: [
        2, 4, 150, 250, 10,
        3, 5, 100, 230, 10,
        5, 25, 5, false
    ],
```

```
    EXTRA_SMALL: [
        2, 2, 150, 250, 10,
        2, 2, 100, 230, 10,
        5, 25, 5
    ],
    EXTRA_SMALL_OPEN: [
        2, 2, 150, 250, 10,
        2, 2, 100, 230, 10,
        5, 25, 5, true
    ],
    EXTRA_SMALL_CLOSED: [
        2, 2, 150, 250, 10,
        2, 2, 100, 230, 10,
        5, 25, 5, false
    ],
    SMALL_OPEN: [
        2, 3, 150, 250, 10,
        2, 4, 100, 230, 10,
        5, 25, 5, true
    ],
    SMALL_CLOSED: [
        2, 3, 150, 250, 10,
        2, 4, 100, 230, 10,
        5, 25, 5, false
    ]
}

jTrainerExt.Participant = class {
    _symbol;
    _potentialSymbol;
    _number;
    _productCount;
    _shippedProductCount;
    _potential;
    _eliminated;

    _maskNumber;
    _maskProductCount;
    _maskRemainingProductCount;
    _maskPotential;

    constructor(symbol, potentialSymbol, number, productCount) {
        this._symbol = symbol;
        this._potentialSymbol = potentialSymbol;
        this._number = number;
        this._productCount = productCount;
        this._shippedProductCount = 0;
        this._eliminated = false;

        this._maskNumber = false;
        this._maskProductCount = false;
        this._maskRemainingProductCount = false;
        this._maskPotential = false;
    }

    create() {
    }

    clone() {
        const clone = this.create();
        clone._symbol = this._symbol;
        clone._potentialSymbol = this._potentialSymbol;
        clone._number = this._number;
```

```
        clone._productCount = this._productCount;
        clone._shippedProductCount = this._shippedProductCount;
        clone._potential = this._potential;
        clone._eliminated = this._eliminated;

        clone._maskNumber = this._maskNumber;
        clone._maskProductCount = this._maskProductCount;
        clone._maskRemainingProductCount = this._maskRemainingProductCount;
        clone._maskPotential = this._maskPotential;
        return clone;
    }

    get symbol() {
        return this._symbol;
    }

    get potentialSymbol() {
        return this._potentialSymbol;
    }

    get number() {
        return this._number;
    }

    set number(number) {
        this._number = number;
    }

    get productCount() {
        return this._productCount;
    }

    set productCount(productCount) {
        this._productCount = productCount;
    }

    get shippedProductCount() {
        return this._shippedProductCount;
    }

    set shippedProductCount(shippedProductCount) {
        this._shippedProductCount = shippedProductCount;
    }

    get remainingProductCount() {
        if (this._productCount === undefined) return undefined;
        return this._productCount - this._shippedProductCount;
    }

    get potential() {
        return this._potential;
    }

    set potential(potential) {
        this._potential = potential;
    }

    get eliminated() {
        return this._eliminated;
    }

    set eliminated(eliminated) {
        this._eliminated = eliminated;
```

```
    }

    get maskNumber() {
        return this._maskNumber;
    }

    set maskNumber(maskNumber) {
        this._maskNumber = maskNumber;
    }

    get maskProductCount() {
        return this._maskProductCount;
    }

    set maskProductCount(maskProductCount) {
        this._maskProductCount = maskProductCount;
    }

    get maskRemainingProductCount() {
        return this._maskRemainingProductCount;
    }

    set maskRemainingProductCount(maskRemainingProductCount) {
        this._maskRemainingProductCount = maskRemainingProductCount;
    }

    get maskPotential() {
        return this._maskPotential;
    }

    set maskPotential(maskPotential) {
        this._maskPotential = maskPotential;
    }

    get defined() {
        return !(this._number === undefined || this._productCount ===
undefined);
    }
}

jTrainerExt.Supplier = class extends jTrainerExt.Participant {
    constructor(number, productCount) {
        super("A", "u", number, productCount);
    }

    create() {
        return new jTrainerExt.Supplier();
    }
}

jTrainerExt.Consumer = class extends jTrainerExt.Participant {
    constructor(number, productCount) {
        super("B", "v", number, productCount);
    }

    create() {
        return new jTrainerExt.Consumer();
    }
}

jTrainerExt.Tariff = class {
    _supplierNumber;
    _consumerNumber;
```

```
    _value;
    _productCount;
    _basic;
    _delta;

    _maskValue;
    _maskProductCount;
    _maskDelta;

    constructor(supplierNumber, consumerNumber, value) {
        this._supplierNumber = supplierNumber;
        this._consumerNumber = consumerNumber;
        this._value = value;
        this._basic = false;

        this._maskValue = false;
        this._maskProductCount = false;
        this._maskDelta = false;
    }

    clone() {
        const clone = new jTrainerExt.Tariff();
        clone._supplierNumber = this._supplierNumber;
        clone._consumerNumber = this._consumerNumber;
        clone._value = this._value;
        clone._productCount = this._productCount;
        clone._basic = this._basic;
        clone._delta = this._delta;

        clone._maskValue = this._maskValue;
        clone._maskProductCount = this._maskProductCount;
        clone._maskDelta = this._maskDelta;
        return clone;
    }

    get supplierNumber() {
        return this._supplierNumber;
    }

    get consumerNumber() {
        return this._consumerNumber;
    }

    get defined() {
        return !(this._supplierNumber === undefined || this._consumerNumber ===
undefined || this._value === undefined);
    }

    eliminated(suppliers, consumers) {
        if (!this.defined) return undefined;
        return suppliers[this._supplierNumber - 1].eliminated ||
consumers[this._consumerNumber - 1].eliminated;
    }

    get value() {
        return this._value;
    }

    set value(value) {
        this._value = value;
    }

    get productCount() {
```

```
        return this._productCount;
    }

    set productCount(value) {
        this._productCount = value;
    }

    get basic() {
        return this._basic;
    }

    set basic(basic) {
        this._basic = basic;
    }

    get delta() {
        return this._delta;
    }

    set delta(value) {
        this._delta = value;
    }

    get maskValue() {
        return this._maskValue;
    }

    set maskValue(maskValue) {
        this._maskValue = maskValue;
    }

    get maskProductCount() {
        return this._maskProductCount;
    }

    set maskProductCount(maskProductCount) {
        this._maskProductCount = maskProductCount;
    }

    get maskDelta() {
        return this._maskDelta;
    }

    set maskDelta(maskDelta) {
        this._maskDelta = maskDelta;
    }
}

jTrainerExt.TransportationProblem = class {
    _suppliers;
    _consumers;
    _tariffs;

    _showPotentials;
    _determined;
    _plan;
    _cycle;
    _cycleNumber;
    _participant;
    _tariff;
    _optimal;
    _shipmentChange;
    _basicCellsForElimination;
```

```
    _tariffsWithMinDelta;

    constructor(suppliers, consumers, tariffs) {
        if (suppliers[0] instanceof jTrainerExt.Supplier) {
            this._suppliers = suppliers;
        } else {
            this._suppliers = suppliers.map((supplierProductCount,
supplierIndex) => new jTrainerExt.Supplier(supplierIndex + 1,
supplierProductCount));
        }

        if (consumers[0] instanceof jTrainerExt.Consumer) {
            this._consumers = consumers;
        } else {
            this._consumers = consumers.map((consumerProductCount,
consumerIndex) => new jTrainerExt.Consumer(consumerIndex + 1,
consumerProductCount));
        }

        if (tariffs[0][0] instanceof jTrainerExt.Tariff) {
            this._tariffs = tariffs;
        } else {
            this._tariffs = tariffs.map((supplierTariffValues, supplierIndex) =>
supplierTariffValues.map((tariffValue, consumerIndex) => new
jTrainerExt.Tariff(supplierIndex + 1, consumerIndex + 1, tariffValue)));
        }
        this._showPotentials = false;
        this._determined = true;
        this._cycleNumber = 0;
        this._optimal = false;
    }

    clone() {
        const clone = new jTrainerExt.TransportationProblem(
            this._suppliers.map(supplier => supplier.clone()),
            this._consumers.map(consumer => consumer.clone()),
            this._tariffs.map(supplierTariffs => supplierTariffs.map(tariff =>
tariff.clone())),
        );
        clone._showPotentials = this._showPotentials;
        clone._determined = this._determined;
        clone._cycleNumber = this._cycleNumber;

        if (this._plan === undefined) {
            clone._plan = this._plan;
        } else {
            clone._plan = [];
            for (let tariffIndex = 0; tariffIndex < this._plan.length;
tariffIndex++) {
                const tariff = this._plan[tariffIndex];
                clone._plan.push(clone._tariffs[tariff.supplierNumber -
1][tariff.consumerNumber - 1]);
            }
        }

        if (this._cycle === undefined) {
            clone._cycle = this._cycle;
        } else {
            clone._cycle = [];
            for (let nodeIndex = 0; nodeIndex < this._cycle.length; nodeIndex++)
{
                const node = this._cycle[nodeIndex];
```

```
                clone._cycle.push(clone._tariffs[node.supplierNumber -
1][node.consumerNumber - 1]);
            }
        }

        if (this._participant instanceof jTrainerExt.Supplier) {
            clone._participant = clone._suppliers[this._participant.number - 1];
        } else if (this._participant instanceof jTrainerExt.Consumer) {
            clone._participant = clone._consumers[this._participant.number - 1];
        }

        if (this._tariff !== undefined) {
            clone._tariff = clone.tariffs[this._tariff.supplierNumber -
1][this._tariff.consumerNumber - 1];
        }

        clone._optimal = this._optimal;

        if (this._tariffsWithMinDelta !== undefined) {
            clone._tariffsWithMinDelta = [];
            for (let tariffIndex = 0; tariffIndex <
this._tariffsWithMinDelta.length; tariffIndex++) {
                const tariff = this._tariffsWithMinDelta[tariffIndex];

clone._tariffsWithMinDelta.push(clone._tariffs[tariff.supplierNumber -
1][tariff.consumerNumber - 1]);
            }
        }

        clone._shipmentChange = this._shipmentChange;

        if (this._basicCellsForElimination !== undefined) {
            clone._basicCellsForElimination = [];
            for (let tariffIndex = 0; tariffIndex <
this._basicCellsForElimination.length; tariffIndex++) {
                const tariff = this._basicCellsForElimination[tariffIndex];

clone._basicCellsForElimination.push(clone._tariffs[tariff.supplierNumber -
1][tariff.consumerNumber - 1]);
            }
        }

        return clone;
    }

    convertToClosed() {
        if (!this.defined) return false;

        const totalProduction = this.totalProduction;
        const totalConsumption = this.totalConsumption;
        const productCount = Math.abs(totalProduction - totalConsumption);

        if (productCount === 0) return false;

        if (totalProduction < totalConsumption) {
            this._suppliers.push(new jTrainerExt.Supplier(this._suppliers.length
+ 1, productCount));
            this._tariffs.push(Array.from({length: this._consumers.length}, (_,
consumerIndex) => new jTrainerExt.Tariff(this._suppliers.length, consumerIndex +
1, 0)));
        } else {
            this._consumers.push(new jTrainerExt.Consumer(this._consumers.length
+ 1, productCount));
```

```
            this._tariffs.forEach((supplierTariffs, supplierIndex) => {
                supplierTariffs.push(new jTrainerExt.Tariff(supplierIndex + 1,
this._consumers.length, 0));
            });
        }
        return true;
    }

    get suppliers() {
        return this._suppliers;
    }

    get consumers() {
        return this._consumers;
    }

    get tariffs() {
        return this._tariffs;
    }

    get defined() {
        return this._suppliers.every(supplier => supplier.defined) &&
            this._consumers.every(consumer => consumer.defined) &&
            this._tariffs.every(supplierTariffs => supplierTariffs.every(tariff
=> tariff.defined));
    }

    get determined() {
        return this._determined;
    }

    set determined(determined) {
        this._determined = determined;
    }

    get plan() {
        return this._plan;
    }

    set plan(plan) {
        this._plan = plan;
    }

    get cycle() {
        return this._cycle;
    }

    set cycle(cycle) {
        this._cycle = cycle;
    }

    get cycleNumber() {
        return this._cycleNumber;
    }

    set cycleNumber(cycleNumber) {
        this._cycleNumber = cycleNumber;
    }

    get totalProduction() {
        const totalProduction = this._suppliers.map(supplier =>
supplier.productCount).reduce((accumulator, currentValue) => accumulator +
currentValue, 0);
```

```
        return isNaN(totalProduction) ? undefined : totalProduction;
    }

    get totalConsumption() {
        const totalConsumption = this._consumers.map(consumer =>
consumer.productCount).reduce((accumulator, currentValue) => accumulator +
currentValue, 0);
        return isNaN(totalConsumption) ? undefined : totalConsumption;
    }

    get open() {
        if (!this.defined) return undefined;
        return this.totalProduction !== this.totalConsumption;
    }

    get remainingSupplierCount() {
        return this._suppliers.map(supplier => supplier.eliminated ? 0 :
1).reduce((accumulator, currentValue) => accumulator + currentValue, 0);
    }

    get remainingConsumerCount() {
        return this._consumers.map(consumer => consumer.eliminated ? 0 :
1).reduce((accumulator, currentValue) => accumulator + currentValue, 0);
    }

    get showPotentials() {
        return this._showPotentials;
    }

    set showPotentials(showPotentials) {
        this._showPotentials = showPotentials;
    }

    get participant() {
        return this._participant;
    }

    set participant(participant) {
        this._participant = participant;
    }

    get tariff() {
        return this._tariff;
    }

    set tariff(tariff) {
        this._tariff = tariff;
    }

    get optimal() {
        return this._optimal;
    }

    set optimal(optimal) {
        this._optimal = optimal;
    }

    get shipmentChange() {
        return this._shipmentChange;
    }

    set shipmentChange(shipmentChange) {
        this._shipmentChange = shipmentChange;
```

```
    }

    get basicCellsForElimination() {
        return this._basicCellsForElimination;
    }

    set basicCellsForElimination(basicCellsForElimination) {
        this._basicCellsForElimination = basicCellsForElimination;
    }

    static #buildCornerCellHTML() {
        return `
      <th class="transportation-table__corner-cell" rowspan="3"
colspan="3"></th>
    `;
    }

    static #buildSupplierPotentialLabelCellHTML() {
        return `
      <th class="transportation-table__potential-label-cell" rowspan="3">U</th>
    `;
    }

    static #buildConsumerPotentialLabelCellHTML() {
        return `
      <th class="transportation-table__potential-label-cell" scope="row"
colspan="3">V</th>
    `;
    }

    static #buildSuppliersLabelHTML(supplierCount) {
        return `
      <th class="transportation-table__suppliers-label" rowspan="${3 *
supplierCount}" scope="row">
        <svg class="transportation-table__suppliers-label-svg" width="40"
height="150">
          <text x="25" y="75" fill="rgb(102, 102, 102)" transform="rotate(-90,
25, 75)" style="text-anchor:middle">Постачальники</text>
        </svg>
      </th>
    `;
    }

    static #buildConsumersLabelHTML(consumerCount) {
        return `
      <th class="transportation-table__consumers-label" colspan="${consumerCount
* 3}">Споживачі</th>
    `;
    }

    static #buildSupplierLabelCellHTML(supplier) {
        const classes = jTrainerExt.calculateClasses({
            "transportation-table__label-cell": true,
            "transportation-table__cell--highlighted": supplier.maskNumber
        });

        return `
      <th ${classes} scope="row" rowspan="3">
        ${supplier.symbol}<sub>${supplier.maskNumber ? "?" :
supplier.number}</sub>
      </th>
    `;
    }
```

```
    static #buildConsumerLabelCellHTML(consumer) {
        const classes = jTrainerExt.calculateClasses({
            "transportation-table__label-cell": true,
            "transportation-table__cell--highlighted": consumer.maskNumber
        });

        return `
      <th ${classes} colspan="3">
        ${consumer.symbol}<sub>${consumer.maskNumber ? "?" :
consumer.number}</sub>
      </th>
    `;
    }

    static #buildSupplierProductCountCellHTML(supplier) {
        const classes = jTrainerExt.calculateClasses({
            "transportation-table__product-count-cell": true,
            "transportation-table__cell--highlighted": supplier.maskProductCount
|| supplier.maskRemainingProductCount,
            "transportation-table__cell--eliminated": supplier.eliminated
        });

        return `
      <td ${classes} rowspan="3">
        ${supplier.productCount !== supplier.remainingProductCount
            ? `<span class="transportation-table__product-count transportation-
table__product-count--changed">
              ${supplier.maskProductCount ? "?" : supplier.productCount}
            </span>
            <span class="transportation-table__remaining-product-count">
              ${supplier.maskRemainingProductCount ? "?" :
supplier.remainingProductCount}
            </span>`
            : `
            <span class="transportation-table__product-count">
              ${supplier.maskProductCount ? "?" :
supplier.remainingProductCount}
            </span>
          `}
      </td>
    `;
    }

    static #buildConsumerProductCountCellHTML(consumer) {
        const classes = jTrainerExt.calculateClasses({
            "transportation-table__product-count-cell": true,
            "transportation-table__cell--highlighted": consumer.maskProductCount
|| consumer.maskRemainingProductCount,
            "transportation-table__cell--eliminated": consumer.eliminated
        });

        return `
      <td ${classes} colspan="3">
        ${consumer.productCount !== consumer.remainingProductCount
            ? `<span class="transportation-table__product-count transportation-
table__product-count--changed">
              ${consumer.maskProductCount ? "?" : consumer.productCount}
            </span>
            <span class="transportation-table__remaining-product-count">
              ${consumer.maskRemainingProductCount ? "?" :
consumer.remainingProductCount}
            </span>`
```

```
            : `
            <span class="transportation-table__product-count">
              ${consumer.maskProductCount ? "?" :
consumer.remainingProductCount}
            </span>
          `}
      </td>
    `;
    }

    static #buildTariffSubcellHTML(tariff) {
        const classes = jTrainerExt.calculateClasses({
            "transportation-table__tariff": true,
            "transportation-table__cell--highlighted": tariff.maskValue
        });

        return `
      <td ${classes}>${tariff.maskValue ? "?" : tariff.value}</td>
    `;
    }

    static #buildDeltaSubcellHTML(tariff, suppliers, consumers) {
        const basic = tariff.basic;
        const delta = tariff.delta;
        const classes = jTrainerExt.calculateClasses({
            "transportation-table__delta": true,
            "transportation-table__subcell--basic": basic && delta ===
undefined,
            "transportation-table__cell--highlighted": !basic &&
tariff.maskDelta,
            "transportation-table__cell--eliminated": !basic && delta ===
undefined && tariff.eliminated(suppliers, consumers)
        });

        return `
      <td ${classes}>${tariff.maskDelta && !basic ? "?" : (delta !== undefined ?
delta.toString().replace("-", "−") : "")}</td>
    `;
    }

    static #buildShipmentSubcellHTML(tariff, suppliers, consumers) {
        const basic = tariff.basic;
        const classes = jTrainerExt.calculateClasses({
            "transportation-table__shipment": true,
            "transportation-table__subcell--basic": basic && tariff.productCount
=== undefined,
            "transportation-table__cell--eliminated": !basic &&
tariff.productCount === undefined && tariff.eliminated(suppliers, consumers),
            "transportation-table__cell--highlighted": basic &&
tariff.productCount !== undefined && tariff.maskProductCount
        });

        return `
      <td ${classes}>${basic && tariff.maskProductCount ? "?" :
(tariff.productCount !== null && tariff.productCount !== undefined ?
tariff.productCount : "")}</td>
    `;
    }

    static #findDirectionBetweenCycleNodes(from, to) {
        if (to.supplierNumber > from.supplierNumber) {
            return jTrainerExt.Direction.BOTTOM;
        } else if (to.consumerNumber > from.consumerNumber) {
```

```
            return jTrainerExt.Direction.RIGHT;
        } else if (to.supplierNumber < from.supplierNumber) {
            return jTrainerExt.Direction.TOP;
        } else if (to.consumerNumber < from.consumerNumber) {
            return jTrainerExt.Direction.LEFT;
        }
        throw new Error("Nodes are the same");
    }

    static #buildSpaceSubcellHTML(tariff, suppliers, consumers,
subcellAlignment, nodes) {
        const classes = jTrainerExt.calculateClasses({
            "transportation-table__space": true,
            "transportation-table__subcell--basic": tariff.basic,
            "transportation-table__cell--eliminated": !tariff.basic &&
tariff.eliminated(suppliers, consumers)
        });

        let content = "";
        if (subcellAlignment !== undefined && nodes !== undefined) {
            // Це фрагмент клітини тарифу, у якому можна малювати й теоретично є
що (цикл або його фрагмент)
            const cyclic = jTrainerExt.isCyclic(nodes);
            const supplierNumber = tariff.supplierNumber;
            const consumerNumber = tariff.consumerNumber;

            let cycleElement = jTrainerExt.CycleElement.NONE;
            let currentNodeIndex = undefined;
            for (let nodeIndex = 0; nodeIndex < nodes.length; nodeIndex++) {
                const node = nodes[nodeIndex];
                if (node === tariff) {
                    cycleElement = jTrainerExt.CycleElement.NODE;
                    currentNodeIndex = nodeIndex;
                    break;
                } else {
                    let nextNode = undefined;
                    if (nodeIndex + 1 < nodes.length) {
                        // Далі ще є вузли в масиві nodes
                        nextNode = nodes[nodeIndex + 1];
                    } else if (cyclic) {
                        // Далі вузлів не було, але завдяки зацикленню
посилаємося на перший вузол
                        nextNode = nodes[0];
                    }

                    if (nextNode) {
                        // Перебираємо парами вузли: 0..1, 1..2, 2..3 -
перевіряємо, чи проходить з'єднання через комірку
                        if (node.supplierNumber === nextNode.supplierNumber &&
                            node.supplierNumber === supplierNumber &&
                            jTrainerExt.isBetween(consumerNumber,
node.consumerNumber, nextNode.consumerNumber)) {
                            // Будуємо горизонтальне з'єднання
                            if (cycleElement ===
jTrainerExt.CycleElement.VERTICAL_LINK) {
                                cycleElement =
jTrainerExt.CycleElement.HORIZONTAL_AND_VERTICAL_LINKS;
                            } else {
                                cycleElement =
jTrainerExt.CycleElement.HORIZONTAL_LINK;
                            }
                        } else if (
                            node.consumerNumber === nextNode.consumerNumber &&
```

```
                            node.consumerNumber === consumerNumber &&
                            jTrainerExt.isBetween(supplierNumber,
node.supplierNumber, nextNode.supplierNumber)) {
                            // Будуємо вертикальне з'єднання
                            if (cycleElement ===
jTrainerExt.CycleElement.HORIZONTAL_LINK) {
                                cycleElement =
jTrainerExt.CycleElement.HORIZONTAL_AND_VERTICAL_LINKS;
                            } else {
                                cycleElement =
jTrainerExt.CycleElement.VERTICAL_LINK;
                            }
                        }
                    }
                }
            }

            // Малюємо в залежності від типу елемента
            if (cycleElement === jTrainerExt.CycleElement.NODE) {
                if (subcellAlignment === jTrainerExt.SubcellAlignment.CENTER) {
                    content = `<div ${jTrainerExt.calculateClasses({
                        "transportation-table__cycle-node": true,
                        "transportation-table__cycle-node--positive":
currentNodeIndex % 2 === 0,
                        "transportation-table__cycle-node--negative":
currentNodeIndex % 2 === 1,
                    })}></div>`;
                } else {
                    let previousDirection;
                    let nextDirection;

                    if (currentNodeIndex - 1 >= 0) {
                        previousDirection =
this.#findDirectionBetweenCycleNodes(tariff, nodes[currentNodeIndex - 1]);
                    } else if (cyclic) {
                        previousDirection =
this.#findDirectionBetweenCycleNodes(tariff, nodes[nodes.length - 1]);
                    }
                    if (currentNodeIndex + 1 < nodes.length) {
                        nextDirection =
this.#findDirectionBetweenCycleNodes(tariff, nodes[currentNodeIndex + 1]);
                    } else if (cyclic) {
                        nextDirection =
this.#findDirectionBetweenCycleNodes(tariff, nodes[0]);
                    }

                    if (
                        (subcellAlignment === jTrainerExt.SubcellAlignment.TOP
&& (previousDirection === jTrainerExt.Direction.TOP || nextDirection ===
jTrainerExt.Direction.TOP)) ||
                        (subcellAlignment ===
jTrainerExt.SubcellAlignment.BOTTOM && (previousDirection ===
jTrainerExt.Direction.BOTTOM || nextDirection === jTrainerExt.Direction.BOTTOM))
                    ) {
                        content = `<div class="transportation-table__cycle-link
transportation-table__cycle-link--vertical"></div>`;
                    } else if (
                        (subcellAlignment === jTrainerExt.SubcellAlignment.LEFT
&& (previousDirection === jTrainerExt.Direction.LEFT || nextDirection ===
jTrainerExt.Direction.LEFT)) ||
                        (subcellAlignment === jTrainerExt.SubcellAlignment.RIGHT
&& (previousDirection === jTrainerExt.Direction.RIGHT || nextDirection ===
jTrainerExt.Direction.RIGHT))
```

```
                    ) {
                        content = `<div class="transportation-table__cycle-link
transportation-table__cycle-link--horizontal"></div>`;
                    }
                }
            } else if (
                cycleElement === jTrainerExt.CycleElement.HORIZONTAL_LINK &&
                (subcellAlignment === jTrainerExt.SubcellAlignment.LEFT ||
subcellAlignment === jTrainerExt.SubcellAlignment.CENTER || subcellAlignment ===
jTrainerExt.SubcellAlignment.RIGHT)
            ) {
                content = `<div class="transportation-table__cycle-link
transportation-table__cycle-link--horizontal"></div>`;
            } else if (
                cycleElement === jTrainerExt.CycleElement.VERTICAL_LINK &&
                (subcellAlignment === jTrainerExt.SubcellAlignment.TOP ||
subcellAlignment === jTrainerExt.SubcellAlignment.CENTER || subcellAlignment ===
jTrainerExt.SubcellAlignment.BOTTOM)
            ) {
                content = `<div class="transportation-table__cycle-link
transportation-table__cycle-link--vertical"></div>`;
            } else if (cycleElement ===
jTrainerExt.CycleElement.HORIZONTAL_AND_VERTICAL_LINKS) {
                if (subcellAlignment === jTrainerExt.SubcellAlignment.CENTER) {
                    content = `<div class="transportation-table__cycle-link-
wrapper">
            <div class="transportation-table__cycle-link transportation-
table__cycle-link--horizontal"></div>
            <div class="transportation-table__cycle-link transportation-
table__cycle-link--vertical"></div>
          </div>`;
                } else if (subcellAlignment ===
jTrainerExt.SubcellAlignment.LEFT || subcellAlignment ===
jTrainerExt.SubcellAlignment.RIGHT) {
                    content = `<div class="transportation-table__cycle-link
transportation-table__cycle-link--horizontal"></div>`;
                } else if (subcellAlignment === jTrainerExt.SubcellAlignment.TOP
|| subcellAlignment === jTrainerExt.SubcellAlignment.BOTTOM) {
                    content = `<div class="transportation-table__cycle-link
transportation-table__cycle-link--vertical"></div>`;
                }
            }
        }

        return `
      <td ${classes}>${content}</td>
    `;
    }

    static #buildSupplierPotentialCellHTML(supplier) {
        const classes = jTrainerExt.calculateClasses({
            "transportation-table__potential-cell": true,
            "transportation-table__cell--highlighted": supplier.maskPotential
        });
        const potential = supplier.maskPotential
            ? "?"
            : (supplier.potential !== undefined ?
supplier.potential.toString().replace("-", "−") : "");

        return `
      <td ${classes} rowspan="3">
        ${supplier.potential !== undefined ?
`${supplier.potentialSymbol}<sub>${supplier.number}</sub> = ${potential}` : ""}
```

```
      </td>
    `;
    }

    static #buildConsumerPotentialCellHTML(consumer) {
        const classes = jTrainerExt.calculateClasses({
            "transportation-table__potential-cell": true,
            "transportation-table__cell--highlighted": consumer.maskPotential
        });
        const potential = consumer.maskPotential
            ? "?"
            : (consumer.potential !== undefined ?
consumer.potential.toString().replace("-", "−") : "");

        return `
      <td ${classes} colspan="3">
        ${consumer.potential !== undefined ?
`${consumer.potentialSymbol}<sub>${consumer.number}</sub> = ${potential}` : ""}
      </td>
    `;
    }

    toHTML() {
        return `
      <table class="transportation-table table table-fit">
        <thead>
          <tr>
            ${jTrainerExt.TransportationProblem.#buildCornerCellHTML()}

${jTrainerExt.TransportationProblem.#buildConsumersLabelHTML(this._consumers.len
gth)}
            ${this._showPotentials ?
jTrainerExt.TransportationProblem.#buildSupplierPotentialLabelCellHTML() : ""}
          </tr>
          <tr>
            ${this._consumers.map(consumer =>
jTrainerExt.TransportationProblem.#buildConsumerLabelCellHTML(consumer)).join("
")}
          </tr>
          <tr>
            ${this._consumers.map(consumer =>
jTrainerExt.TransportationProblem.#buildConsumerProductCountCellHTML(consumer)).
join("")}
          </tr>
        </thead>
        <tbody>
          ${this._suppliers.map((supplier, supplierIndex) => `
            <tr class="transportation-table__row">
              ${supplierIndex === 0 ?
jTrainerExt.TransportationProblem.#buildSuppliersLabelHTML(this._suppliers.lengt
h) : ""}

${jTrainerExt.TransportationProblem.#buildSupplierLabelCellHTML(supplier)}

${jTrainerExt.TransportationProblem.#buildSupplierProductCountCellHTML(supplier)
}
              ${this._consumers.map((consumer, consumerIndex) => `

${jTrainerExt.TransportationProblem.#buildTariffSubcellHTML(this._tariffs[suppli
erIndex][consumerIndex])}
```

```
${jTrainerExt.TransportationProblem.#buildSpaceSubcellHTML(this._tariffs[supplie
rIndex][consumerIndex], this._suppliers, this._consumers,
jTrainerExt.SubcellAlignment.TOP, this._cycle)}

${jTrainerExt.TransportationProblem.#buildDeltaSubcellHTML(this._tariffs[supplie
rIndex][consumerIndex], this._suppliers, this._consumers)}
              `).join("")}
              ${this._showPotentials ?
jTrainerExt.TransportationProblem.#buildSupplierPotentialCellHTML(supplier) :
""}
            </tr>
            <tr class="transportation-table__row">
              ${this._consumers.map((consumer, consumerIndex) => `

${jTrainerExt.TransportationProblem.#buildSpaceSubcellHTML(this._tariffs[supplie
rIndex][consumerIndex], this._suppliers, this._consumers,
jTrainerExt.SubcellAlignment.LEFT, this._cycle)}

${jTrainerExt.TransportationProblem.#buildSpaceSubcellHTML(this._tariffs[supplie
rIndex][consumerIndex], this._suppliers, this._consumers,
jTrainerExt.SubcellAlignment.CENTER, this._cycle)}

${jTrainerExt.TransportationProblem.#buildSpaceSubcellHTML(this._tariffs[supplie
rIndex][consumerIndex], this._suppliers, this._consumers,
jTrainerExt.SubcellAlignment.RIGHT, this._cycle)}
              `).join("")}
            </tr>
            <tr class="transportation-table__row">
              ${this._consumers.map((consumer, consumerIndex) => `

${jTrainerExt.TransportationProblem.#buildSpaceSubcellHTML(this._tariffs[supplie
rIndex][consumerIndex], this._suppliers, this._consumers, undefined,
this._cycle)}

${jTrainerExt.TransportationProblem.#buildSpaceSubcellHTML(this._tariffs[supplie
rIndex][consumerIndex], this._suppliers, this._consumers,
jTrainerExt.SubcellAlignment.BOTTOM, this._cycle)}

${jTrainerExt.TransportationProblem.#buildShipmentSubcellHTML(this._tariffs[supp
lierIndex][consumerIndex], this._suppliers, this._consumers)}
              `).join("")}
            </tr>
          `).join("")}
          ${this._showPotentials ? `
            <tr class="transportation-table__row">

${jTrainerExt.TransportationProblem.#buildConsumerPotentialLabelCellHTML()}
              ${this._consumers.map(consumer =>
jTrainerExt.TransportationProblem.#buildConsumerPotentialCellHTML(consumer)).joi
n("")}
            </tr>
          ` : ""}
        </tbody>
      </table>
    `;
    }

    static generate(
        minSupplierCount, maxSupplierCount, minSupplierProductCount,
maxSupplierProductCount, supplierProductCountStep,
        minConsumerCount, maxConsumerCount, minConsumerProductCount,
maxConsumerProductCount, consumerProductCountStep,
```

```
        minTariffValue, maxTariffValue, tariffValueStep, open = undefined
    ) {
        let transportationProblem;
        do {
            const suppliers = jTrainerExt.generateSuppliers(minSupplierCount,
maxSupplierCount, minSupplierProductCount, maxSupplierProductCount,
supplierProductCountStep);
            const consumers = jTrainerExt.generateConsumers(minConsumerCount,
maxConsumerCount, minConsumerProductCount, maxConsumerProductCount,
consumerProductCountStep);
            const tariffs = jTrainerExt.generateTariffs(suppliers.length,
consumers.length, minTariffValue, maxTariffValue, tariffValueStep);
            transportationProblem = new
jTrainerExt.TransportationProblem(suppliers, consumers, tariffs);

            // Повторюємо лише тоді, коли:
            // open !== undefined - тоді нам важливий тип (відкритий чи
закритий) і задача йому не відповідає
        } while (open !== undefined && transportationProblem.open !== open);
        return transportationProblem;
    }

    static solve(transportationProblem,
                 stopConditionChecker = () => false,
                 clone = true,
                 determined = true,
                 basicPlanConstructionMethod =
jTrainerExt.BasicPlanConstructionMethod.LEAST_COST) {
        // determined - параметр, визначає, чи буде задача розв'язана
передбачуваним способом
        // solvedTransportationProblem.determined - чи є процес розв'язання
варіативним сам по собі
        const solvedTransportationProblem = clone ?
transportationProblem.clone() : transportationProblem;
        if (stopConditionChecker(solvedTransportationProblem)) return
solvedTransportationProblem;

        // Якщо транспортна задача відкритого типу - виконуємо приведення до
закритого
        solvedTransportationProblem.convertToClosed();
        if (stopConditionChecker(solvedTransportationProblem)) return
solvedTransportationProblem;

        // Побудова опорного плану за обраним методом. Якщо не вказаний -
обираємо випадковий. Якщо детерміновано -
        // конкретний (метод мінімальної вартості).
        solvedTransportationProblem.plan = [];
        if (
            basicPlanConstructionMethod === undefined ||
            (basicPlanConstructionMethod !==
jTrainerExt.BasicPlanConstructionMethod.LEAST_COST &&
basicPlanConstructionMethod !==
jTrainerExt.BasicPlanConstructionMethod.NORTHWEST_CORNER)
        ) {
            basicPlanConstructionMethod = determined
                ? jTrainerExt.BasicPlanConstructionMethod.LEAST_COST
                : (Math.random() < 0.5 ?
jTrainerExt.BasicPlanConstructionMethod.LEAST_COST :
jTrainerExt.BasicPlanConstructionMethod.NORTHWEST_CORNER);
        }

        let remainingSupplierCount;
        let remainingConsumerCount;
```

```
        while (
            (
                remainingSupplierCount =
solvedTransportationProblem.remainingSupplierCount,
                    remainingConsumerCount =
solvedTransportationProblem.remainingConsumerCount, true
            ) && (remainingSupplierCount > 0 || remainingConsumerCount > 0)
            ) {
            if (stopConditionChecker(solvedTransportationProblem)) return
solvedTransportationProblem;

            // Вибір наступної клітини для виконання поставки залежить від
методу й від того, які учасники вибули
            let tariff;
            if (basicPlanConstructionMethod ===
jTrainerExt.BasicPlanConstructionMethod.LEAST_COST) {
                const sortedTariffs = [];
                jTrainerExt.forEachTariff(solvedTransportationProblem.tariffs,
tariff => {
                    if
(!tariff.eliminated(solvedTransportationProblem.suppliers,
solvedTransportationProblem.consumers)) {
                        sortedTariffs.push(tariff);
                    }
                });

                sortedTariffs.sort((a, b) => {
                    if (a.value === 0) return 1;
                    if (b.value === 0) return -1;
                    return a.value - b.value;
                });
                tariff = sortedTariffs[0];

                if (sortedTariffs.length > 1 && sortedTariffs[0].value ===
sortedTariffs[1].value) {
                    solvedTransportationProblem.determined = false;
                }
            } else {
                tariff = solvedTransportationProblem.tariffs
                    [solvedTransportationProblem.suppliers.find(supplier =>
!supplier.eliminated).number - 1]
                    [solvedTransportationProblem.consumers.find(consumer =>
!consumer.eliminated).number - 1];
            }

            // Виконання поставки
            let supplier =
solvedTransportationProblem.suppliers[tariff.supplierNumber - 1];
            let consumer =
solvedTransportationProblem.consumers[tariff.consumerNumber - 1];
            const shipment = Math.min(supplier.remainingProductCount,
consumer.remainingProductCount);

            solvedTransportationProblem.plan.push(tariff);
            if (stopConditionChecker(solvedTransportationProblem)) return
solvedTransportationProblem;

            tariff.productCount = shipment;
            tariff.basic = true;
            supplier.shippedProductCount += shipment;
            consumer.shippedProductCount += shipment;
            if (stopConditionChecker(solvedTransportationProblem)) return
solvedTransportationProblem;
```

```
            // Видалення постачальника та/або споживача: порівнюємо залишки
            if (
                supplier.remainingProductCount === 0 &&
                consumer.remainingProductCount === 0
            ) {
                // Запаси постачальника рівні потребам споживача - треба
визначитися, кого з них видаляти
                if (remainingSupplierCount === 1 && remainingConsumerCount ===
1) {
                    // Клітина тарифу була останньою - видаляємо й
постачальника, й споживача
                    supplier.eliminated = true;
                    consumer.eliminated = true;
                } else {
                    // Клітина не остання - треба видалити або постачальника,
або споживача (на вибір); потім - зробити
                    // клітину базисною для постачальника або споживача, що
залишиться (й видалити його)
                    solvedTransportationProblem.determined = false;

                    let otherTariff;
                    if (determined || Math.random() < 0.5) {
                        supplier.eliminated = true;

                        const supplierIndices = [];
                        for (let supplierIndex = 0; supplierIndex <
solvedTransportationProblem.suppliers.length; supplierIndex++) {
                            if
(!solvedTransportationProblem.tariffs[supplierIndex][consumer.number -
1].eliminated(solvedTransportationProblem.suppliers,
solvedTransportationProblem.consumers)) {
                                supplierIndices.push(supplierIndex);
                            }
                        }

                        const index = determined ? supplierIndices[0] :
jTrainerExt.getRandomElement(supplierIndices);
                        otherTariff =
solvedTransportationProblem.tariffs[index][consumer.number - 1];
                        consumer.eliminated = true;
                    } else {
                        consumer.eliminated = true;

                        const consumerIndices = [];
                        for (let consumerIndex = 0; consumerIndex <
solvedTransportationProblem.consumers.length; consumerIndex++) {
                            if
(!solvedTransportationProblem.tariffs[supplier.number -
1][consumerIndex].eliminated(solvedTransportationProblem.suppliers,
solvedTransportationProblem.consumers)) {
                                consumerIndices.push(consumerIndex);
                            }
                        }

                        const index = determined ? consumerIndices[0] :
jTrainerExt.getRandomElement(consumerIndices);
                        otherTariff =
solvedTransportationProblem.tariffs[supplier.number - 1][index];
                        supplier.eliminated = true;
                    }

                    // 0 базисна клітина
```

```
                    otherTariff.productCount = 0;
                    otherTariff.basic = true;
                }
            } else {
                // Відповідність лише одна: буде видалений або постачальник, або
споживач
                if (supplier.remainingProductCount === 0) {
                    supplier.eliminated = true;
                } else {
                    consumer.eliminated = true;
                }
            }
        }

        if (
            solvedTransportationProblem.suppliers.length === 1 ||
            solvedTransportationProblem.consumers.length === 1
        ) {
            return solvedTransportationProblem;
        }
        if (stopConditionChecker(solvedTransportationProblem)) return
solvedTransportationProblem;

        // Застосування методу потенціалів
        solvedTransportationProblem.showPotentials = true;
        if (stopConditionChecker(solvedTransportationProblem)) return
solvedTransportationProblem;

        while (!solvedTransportationProblem.optimal) {
            // Скидання потенціалів, визначених раніше
            for (const supplier of solvedTransportationProblem.suppliers) {
                supplier.potential = undefined;
            }
            for (const consumer of solvedTransportationProblem.consumers) {
                consumer.potential = undefined;
            }
            solvedTransportationProblem.participant = undefined;
            if (stopConditionChecker(solvedTransportationProblem)) return
solvedTransportationProblem;

            // Скидання значень оцінок, визначених раніше
            jTrainerExt.forEachTariff(solvedTransportationProblem, tariff =>
tariff.delta = undefined);
            if (stopConditionChecker(solvedTransportationProblem)) return
solvedTransportationProblem;

            // Визначення першого потенціалу випадковим чином. Для зручності -
починаємо з 0.
            if (determined) {
                solvedTransportationProblem.suppliers[0].potential = 0;
                solvedTransportationProblem.participant =
solvedTransportationProblem.suppliers[0];
            } else {
                const participant =
jTrainerExt.getRandomElement(solvedTransportationProblem[Math.random() < 0.5 ?
"suppliers" : "consumers"]);
                participant.potential = 0;
                solvedTransportationProblem.participant = participant;
            }
            if (stopConditionChecker(solvedTransportationProblem)) return
solvedTransportationProblem;

            // Перебираємо базисні клітини, поки не знайдемо всі потенціали
```

```
            while (
                solvedTransportationProblem.suppliers.some(supplier =>
supplier.potential === undefined) ||
                solvedTransportationProblem.consumers.some(consumer =>
consumer.potential === undefined)
                ) {
                for (let supplierIndex = 0; supplierIndex <
solvedTransportationProblem.suppliers.length; supplierIndex++) {
                    for (let consumerIndex = 0; consumerIndex <
solvedTransportationProblem.consumers.length; consumerIndex++) {
                        const tariff =
solvedTransportationProblem.tariffs[supplierIndex][consumerIndex];
                        if (tariff.basic) {
                            if (

solvedTransportationProblem.suppliers[supplierIndex].potential !== undefined &&

solvedTransportationProblem.consumers[consumerIndex].potential === undefined
                            ) {

solvedTransportationProblem.consumers[consumerIndex].potential = tariff.value -
solvedTransportationProblem.suppliers[supplierIndex].potential;
                                solvedTransportationProblem.participant =
solvedTransportationProblem.consumers[consumerIndex];
                                if
(stopConditionChecker(solvedTransportationProblem)) return
solvedTransportationProblem;
                            } else if (

solvedTransportationProblem.suppliers[supplierIndex].potential === undefined &&

solvedTransportationProblem.consumers[consumerIndex].potential !== undefined
                            ) {

solvedTransportationProblem.suppliers[supplierIndex].potential = tariff.value -
solvedTransportationProblem.consumers[consumerIndex].potential;
                                solvedTransportationProblem.participant =
solvedTransportationProblem.suppliers[supplierIndex];
                                if
(stopConditionChecker(solvedTransportationProblem)) return
solvedTransportationProblem;
                            }
                        }
                    }
                }
            }

            // Визначимо значення оцінок для вільних клітин
            let minDelta = undefined;
            for (let supplierIndex = 0; supplierIndex <
solvedTransportationProblem.suppliers.length; supplierIndex++) {
                for (let consumerIndex = 0; consumerIndex <
solvedTransportationProblem.consumers.length; consumerIndex++) {
                    const tariff =
solvedTransportationProblem.tariffs[supplierIndex][consumerIndex];
                    if (!tariff.basic) {
                        tariff.delta = tariff.value -
(solvedTransportationProblem.suppliers[supplierIndex].potential +
solvedTransportationProblem.consumers[consumerIndex].potential);
                        solvedTransportationProblem.tariff = tariff;
                        if (stopConditionChecker(solvedTransportationProblem))
return solvedTransportationProblem;
```

```
                        if (minDelta === undefined || tariff.delta < minDelta) {
                            minDelta = tariff.delta;
                        }
                    }
                }
            }

            solvedTransportationProblem.tariffsWithMinDelta = [];
            if (stopConditionChecker(solvedTransportationProblem)) return
solvedTransportationProblem;
            jTrainerExt.forEachTariff(solvedTransportationProblem.tariffs,
tariff => {
                if (tariff.basic) return;

                if (tariff.delta === minDelta) {

solvedTransportationProblem.tariffsWithMinDelta.push(tariff);
                }
            });
            if (stopConditionChecker(solvedTransportationProblem)) return
solvedTransportationProblem;

            let tariffWithMinDelta = determined ?
solvedTransportationProblem.tariffsWithMinDelta[0] :
jTrainerExt.getRandomElement(solvedTransportationProblem.tariffsWithMinDelta);
            if (solvedTransportationProblem.tariffsWithMinDelta.length > 1) {
                solvedTransportationProblem.determined = false;
            }

            if (tariffWithMinDelta.delta >= 0) {
                solvedTransportationProblem.optimal = true;
                if (stopConditionChecker(solvedTransportationProblem)) return
solvedTransportationProblem;
            } else {
                // Треба побудувати цикл переходу й застосувати його. Ідея -
переходити між клітинами, кожен раз
                // змінюючи напрямок на 90 градусів. Якщо глухий кут - пошук
припиняється.
                let cycle = jTrainerExt.buildCycle([tariffWithMinDelta],
undefined, solvedTransportationProblem.suppliers,
solvedTransportationProblem.consumers, solvedTransportationProblem.tariffs);
                solvedTransportationProblem.cycle = cycle;
                solvedTransportationProblem.cycleNumber++;
                if (stopConditionChecker(solvedTransportationProblem)) return
solvedTransportationProblem;

                // Визначення об'єму перерозподілу (за від'ємними вузлами)
                solvedTransportationProblem.shipmentChange = undefined;
                for (let nodeIndex = 1; nodeIndex < cycle.length; nodeIndex +=
2) {
                    if (solvedTransportationProblem.shipmentChange === undefined
|| cycle[nodeIndex].productCount < solvedTransportationProblem.shipmentChange) {
                        solvedTransportationProblem.shipmentChange =
cycle[nodeIndex].productCount;
                    }
                }
                if (stopConditionChecker(solvedTransportationProblem)) return
solvedTransportationProblem;

                solvedTransportationProblem.basicCellsForElimination = [];
                for (let nodeIndex = 1; nodeIndex < cycle.length; nodeIndex +=
2) {
```

```
                    if (cycle[nodeIndex].productCount ===
solvedTransportationProblem.shipmentChange) {

solvedTransportationProblem.basicCellsForElimination.push(cycle[nodeIndex]);
                    }
                }
                if (stopConditionChecker(solvedTransportationProblem)) return
solvedTransportationProblem;

                // Введення до базису нової клітини
                cycle[0].basic = true;
                cycle[0].delta = undefined;
                cycle[0].productCount = 0;

                // Застосування циклу переходу
                for (let nodeIndex = 0; nodeIndex < cycle.length; nodeIndex++) {
                    cycle[nodeIndex].productCount +=
solvedTransportationProblem.shipmentChange * (nodeIndex % 2 === 0 ? 1 : -1);
                }

                // Видалення випадково обраної базисної клітини
                const tariffForElimination = determined ?
solvedTransportationProblem.basicCellsForElimination[0] :
jTrainerExt.getRandomElement(solvedTransportationProblem.basicCellsForEliminatio
n);
                tariffForElimination.basic = false;
                tariffForElimination.productCount = undefined;

                if (solvedTransportationProblem.basicCellsForElimination.length
> 1) {
                    solvedTransportationProblem.determined = false;
                }
            }
        }

        return solvedTransportationProblem;
    }
}

jTrainerExt.getRandomIntInRange = function (min, max, step = 1) {
    const rangeSize = Math.floor((max - min) / step) + 1;
    const randomIndex = Math.floor(Math.random() * rangeSize);
    return min + randomIndex * step;
}

jTrainerExt.getRandomElement = function (array) {
    return array[jTrainerExt.getRandomIntInRange(0, array.length - 1)];
}

jTrainerExt.generateSuppliers = function (minSupplierCount, maxSupplierCount,
minProductCount, maxProductCount, productCountStep) {
    return Array.from({length: jTrainerExt.getRandomIntInRange(minSupplierCount,
maxSupplierCount)}, (_, supplierIndex) =>
        new jTrainerExt.Supplier(supplierIndex + 1,
jTrainerExt.getRandomIntInRange(minProductCount, maxProductCount,
productCountStep)));
}

jTrainerExt.generateConsumers = function (minConsumerCount, maxConsumerCount,
minProductCount, maxProductCount, productCountStep) {
    return Array.from({length: jTrainerExt.getRandomIntInRange(minConsumerCount,
maxConsumerCount)}, (_, consumerIndex) =>
```

```
        new jTrainerExt.Consumer(consumerIndex + 1,
jTrainerExt.getRandomIntInRange(minProductCount, maxProductCount,
productCountStep)));
}

jTrainerExt.generateTariffs = function (supplierCount, consumerCount,
minTariffValue, maxTariffValue, tariffValueStep) {
    return Array.from({length: supplierCount}, (_, supplierIndex) =>
        Array.from({length: consumerCount}, (_, consumerIndex) => new
jTrainerExt.Tariff(supplierIndex + 1, consumerIndex + 1,
jTrainerExt.getRandomIntInRange(minTariffValue, maxTariffValue,
tariffValueStep))));
}

jTrainerExt.forEachTariff = function (tariffs, callback) {
    for (let supplierIndex = 0; supplierIndex < tariffs.length; supplierIndex++)
{
        for (let consumerIndex = 0; consumerIndex <
tariffs[supplierIndex].length; consumerIndex++) {
            callback(tariffs[supplierIndex][consumerIndex], supplierIndex,
consumerIndex);
        }
    }
}

jTrainerExt.calculateClasses = function (object) {
    let classes = "";
    for (const className in object) {
        if (object[className]) {
            classes += (classes !== "" ? " " : "") + className;
        }
    }
    return `class="${classes}"`;
}

jTrainerExt.tryToAnnihilateEntry = function (set, entry, oppositeEntry) {
    if (set.has(oppositeEntry)) {
        set.delete(oppositeEntry);
    } else {
        set.add(entry);
    }
}

jTrainerExt.isCyclic = function (nodes) {
    const data = new Set();
    for (let nodeIndex = 0; nodeIndex < nodes.length; nodeIndex++) {
        const sign = nodeIndex % 2 === 0 ? "+" : "-";
        const oppositeSign = sign === "+" ? "-" : "+";

        jTrainerExt.tryToAnnihilateEntry(data,
`${sign}s${nodes[nodeIndex].supplierNumber}`,
`${oppositeSign}s${nodes[nodeIndex].supplierNumber}`);
        jTrainerExt.tryToAnnihilateEntry(data,
`${sign}c${nodes[nodeIndex].consumerNumber}`,
`${oppositeSign}c${nodes[nodeIndex].consumerNumber}`);
    }
    return data.size === 0;
}

jTrainerExt.buildCycle = function (nodes, previousDirection, suppliers,
consumers, tariffs) {
    // Перевірка циклу на завершеність побудови
    if (jTrainerExt.isCyclic(nodes)) {
```

```
        return nodes;
    }

    // Перебираємо всі можливі варіанти продовження циклу (додавання вузла):
шукаємо базисні клітини відносно останньої
    // в усіх напрямках, крім попереднього
    const currentNode = nodes[nodes.length - 1];
    for (const directionName in jTrainerExt.Direction) {
        const currentDirection = jTrainerExt.Direction[directionName];
        if (currentDirection !== previousDirection) {
            if (currentDirection === jTrainerExt.Direction.RIGHT &&
previousDirection !== jTrainerExt.Direction.LEFT) {
                for (let consumerIndex = currentNode.consumerNumber;
consumerIndex < consumers.length; consumerIndex++) {
                    const tariff = tariffs[currentNode.supplierNumber -
1][consumerIndex];
                    if (tariff.basic && !nodes.includes(tariff)) {
                        const extendedNodes = jTrainerExt.buildCycle([...nodes,
tariff], currentDirection, suppliers, consumers, tariffs);
                        if (extendedNodes !== undefined) return extendedNodes;
                    }
                }
            } else if (currentDirection === jTrainerExt.Direction.TOP &&
previousDirection !== jTrainerExt.Direction.BOTTOM) {
                for (let supplierIndex = currentNode.supplierNumber - 2;
supplierIndex >= 0; supplierIndex--) {
                    const tariff =
tariffs[supplierIndex][currentNode.consumerNumber - 1];
                    if (tariff.basic && !nodes.includes(tariff)) {
                        const extendedNodes = jTrainerExt.buildCycle([...nodes,
tariff], currentDirection, suppliers, consumers, tariffs);
                        if (extendedNodes !== undefined) return extendedNodes;
                    }
                }
            } else if (currentDirection === jTrainerExt.Direction.LEFT &&
previousDirection !== jTrainerExt.Direction.RIGHT) {
                for (let consumerIndex = currentNode.consumerNumber - 2;
consumerIndex >= 0; consumerIndex--) {
                    const tariff = tariffs[currentNode.supplierNumber -
1][consumerIndex];
                    if (tariff.basic && !nodes.includes(tariff)) {
                        const extendedNodes = jTrainerExt.buildCycle([...nodes,
tariff], currentDirection, suppliers, consumers, tariffs);
                        if (extendedNodes !== undefined) return extendedNodes;
                    }
                }
            } else if (currentDirection === jTrainerExt.Direction.BOTTOM &&
previousDirection !== jTrainerExt.Direction.TOP) {
                for (let supplierIndex = currentNode.supplierNumber;
supplierIndex < suppliers.length; supplierIndex++) {
                    const tariff =
tariffs[supplierIndex][currentNode.consumerNumber - 1];
                    if (tariff.basic && !nodes.includes(tariff)) {
                        const extendedNodes = jTrainerExt.buildCycle([...nodes,
tariff], currentDirection, suppliers, consumers, tariffs);
                        if (extendedNodes !== undefined) return extendedNodes;
                    }
                }
            }
        }
    }

    // Якщо шлях виявився глухим кутом - повертаємо undefined
```

```
    return undefined;
}

jTrainerExt.isBetween = function (number, a, b) {
    return (number >= a && number <= b) || (number >= b && number <= a);
}

jTrainerExt.generateUntil = function (predicate, generator) {
    let generatedValue;
    do {
        generatedValue = generator();
    } while (!predicate(generatedValue));
    return generatedValue;
}

jTrainerExt.pad = function (num, size) {
    num = num.toString();
    while (num.length < size) num = "0" + num;
    return num;
}

jTrainerExt.timeFormat = function (timeAsString) {
    return timeAsString.split(":").map(t => jTrainerExt.pad(t, 2)).join(":");
}

jTrainerExt.disable = function (jQueryElement) {
    jQueryElement.attr('disabled', 'disabled').addClass('disabled');
}

jTrainerExt.enable = function (jQueryElement) {
    jQueryElement.removeAttr('disabled').removeClass('disabled');
}

jTrainerExt.shuffleArray = function (array) {
    let index = array.length;
    let randomIndex;

    while (index !== 0) {
        randomIndex = Math.floor(Math.random() * index);
        index--;
        [array[index], array[randomIndex]] = [array[randomIndex], array[index]];
    }

    return array;
}

jTrainerExt.shuffleChildElements = function (parentElement) {
    const childElements = jTrainerExt.shuffleArray([...parentElement.children]);
    void parentElement.offsetWidth;
    for (const childElement of childElements) {
        parentElement.appendChild(childElement);
    }
}

jTrainerExt.replaceHTMLPlaceholders = function (element, placeholders) {
    let innerHTML = element.innerHTML;
    for (const placeholderName in placeholders) {
        innerHTML = innerHTML.replace(`{{{${placeholderName}}}}`,
placeholders[placeholderName]);
    }
    element.innerHTML = innerHTML;
}
```

```
jTrainerExt.validate = function (validator, jQueryButton) {
    validator.setAttemptsOnCheckButton(jQueryButton);
    if (validator.validate() || validator.getAttempts() === 0) {
        jTrainerExt.disable(jQueryButton);
        jQueryButton.parent().parent().find('input').each(function () {
            jTrainerExt.disable($(this));
        });
        jTrainerExt.disable($('#endTrainer'));
    }
}
```

ДОДАТОК К

ПРОГРАМНА РЕАЛІЗАЦІЯ ІНСТРУМЕНТІВ ОСВІТНЬОГО ЕКСПЕРИМЕНТУ

К.1 Інструмент для збору результатів на платформі Mix

```
function getResultsPageURL(courseId, taskId, userId) {
  return
`https://mix.sumdu.edu.ua/teach_class/results/${courseId}/by_task_user/${taskId}
?user_id=${userId}`;
}

function parseDate(formattedDate) {
  const regex = /(-?\d+)\.(\d{2})\.(\d{2}) (\d{2}):(\d{2})/;
  const match = regex.exec(formattedDate);

  if (!match) return undefined;
  return new Date(
    parseInt(match[1]),
    parseInt(match[2]) - 1,
    parseInt(match[3]),
    parseInt(match[4]),
    parseInt(match[5])
  );
}

function formatDate(date) {
  return `${date.getFullYear().toString().padStart(2, "0")}.${(date.getMonth() +
1).toString().padStart(2, "0")}.${date.getDate().toString().padStart(2, "0")}
${date.getHours().toString().padStart(2,
"0")}:${date.getMinutes().toString().padStart(2, "0")}`;
}

async function fetchUserResults(courseId, taskId, userId) {
  const response = await fetch(getResultsPageURL(courseId, taskId, userId));
  const loadedDocument = new DOMParser().parseFromString(await response.text(),
"text/html");
  const rowElement = loadedDocument.querySelector(".col.s12.m12.l9 > .card-
panel:nth-child(2) table > tbody > tr:nth-child(2)");

  return {
    startDate: parseDate(rowElement.querySelector("td:nth-child(2)").innerText),
    finishDate: parseDate(rowElement.querySelector("td:nth-
child(3)").innerText),
  };
}

async function fetchFormattedResults(courseId, names, nodeId) {
  let results = "";

  const task = m_all_tasks.find(task => task["node_id"] === nodeId);
  if (!task) return results;

  for (let i = 0; i < names.length; i++) {
    const user = m_students.find(user => user.name === names[i]);
```

```
    if (user) {
      let scorePercent = m_result_matrix[`usr_${user.id}`]?.[`n_${nodeId}`];
      if (scorePercent === 0) {
        scorePercent = undefined;
      }
      if (scorePercent !== undefined) {
        scorePercent = (scorePercent / 100).toString().replace(".", ",");
      }

      if (scorePercent !== undefined) {
        const userResults = await fetchUserResults(courseId, task["pt_id"],
user.id);
        results +=
`${formatDate(userResults.startDate)}\t${formatDate(userResults.finishDate)}\t${
scorePercent}\n`;
      } else {
        results += "\t\t\n";
      }
    } else {
      results += "\t\t\n";
    }
  }

  return results;
}
```

К.2 Сценарій консолідації даних учасників експерименту до спільної таблиці

```
import pandas as pd
from datetime import datetime


# Часові рамки експерименту
test_1_datetime_from = datetime(2024, 2, 16)
test_1_datetime_to = datetime(2024, 2, 17)
test_2_datetime_from = datetime(2024, 2, 23)
test_2_datetime_to = datetime(2024, 2, 24)

# Завантаження таблиць: дані з платформи Mix та Google Forms
df = pd.read_csv("resources/participants.csv", delimiter=";")
survey = pd.concat([
    pd.read_csv("resources/Анкета учасника освітнього експерименту (варіант
I).csv"),
    pd.read_csv("resources/Анкета учасника освітнього експерименту (варіант
II).csv")
], ignore_index=True)

# Підготовка стовпчиків таблиці з результатами анкетування до консолідації
survey.rename(columns={
    "Отметка времени": "survey_finish_timestamp",
    "Прізвище, ім'я, по-батькові": "full_name",
    "Як Ви оцінюєте вплив роботи з тренажером на Ваш рівень мотивації до
навчання?": f"survey_answer_1",
    "Чи згодні Ви з тим, що запропонований тренажер повноцінно моделює процес
розв'язання транспортних задач?": f"survey_answer_2"
}, inplace=True)

# Очищення даних
```

```
survey["full_name"] = survey["full_name"].str.strip()

# Консолідація
df = df.merge(survey.iloc[:, [0, 1, 2, 3]], on="full_name", how="left")

# Перетворення даних
df["survey_answer_1_points"] = df["survey_answer_1"].map({
    "Виключно негативний": -2,
    "Переважно негативний": -1,
    "Нейтральний": 0,
    "Переважно позитивний": 1,
    "Виключно позитивний": 2,
})
df["survey_answer_2_points"] = df["survey_answer_2"].map({
    "Повністю не погоджуюся": -2,
    "Частково не погоджуюся": -1,
    "Не знаю": 0,
    "Частково погоджуюся": 1,
    "Повністю погоджуюся": 2,
})

# Встановлення типів даних (для міток часу)
for column_name in (
        "test_1_start_timestamp",
        "test_1_finish_timestamp",
        "trainer_last_finish_timestamp",
        "test_2_start_timestamp",
        "test_2_finish_timestamp",
):
    df[column_name] = pd.to_datetime(df[column_name])
df["survey_finish_timestamp"] = pd.to_datetime(df["survey_finish_timestamp"],
errors="coerce")
df["survey_finish_timestamp"] = df["survey_finish_timestamp"].dt.strftime("%Y-
%m-%dT%H:%M:%S")

# Створення допоміжних стовпчиків
df["test_1_duration"] = (df["test_1_finish_timestamp"] -
df["test_1_start_timestamp"]).dt.total_seconds() / 60
df["test_2_duration"] = (df["test_2_finish_timestamp"] -
df["test_2_start_timestamp"]).dt.total_seconds() / 60

df["test_1_requirements_met"] =(
    df["test_1_start_timestamp"].apply(
        lambda timestamp: test_1_datetime_from < timestamp <
test_1_datetime_to))

df["trainer_requirements_met"] = \
    (df["test_1_requirements_met"]) & \
    (df["trainer_score_percentage"] >= 60) & \
    df["trainer_last_finish_timestamp"].apply(
        lambda timestamp: test_1_datetime_to < timestamp < test_2_datetime_from)

df["test_2_requirements_met"] = \
    (df["trainer_requirements_met"]) & \
    (df["trainer_last_finish_timestamp"] < df["test_2_start_timestamp"]) & \
    df["test_2_start_timestamp"].apply(
        lambda timestamp: test_2_datetime_from < timestamp < test_2_datetime_to)

df["requirements_met"] = \
    (df["test_2_requirements_met"]) & \
    (df["survey_finish_timestamp"].notnull()) & \
    (df["test_2_finish_timestamp"] < df["survey_finish_timestamp"])
```

```
df["duration_progress"] = df["test_1_duration"] - df["test_2_duration"]
df["score_percentage_progress"] = (df["test_2_score_percentage"] -
df["test_1_score_percentage"]) / (100 - df["test_1_score_percentage"])

# Збереження результату
df.to_csv("data.csv", sep=";", index=False)
```

К.3 Спільний код для статистичного аналізу ознак

```
import math
import os
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from scipy.stats import ttest_ind, shapiro, mannwhitneyu
from scipy.stats import t, f

default_alpha = 0.05
default_precision = 3


def get_data():
    data_path = os.path.join(os.path.dirname(os.path.dirname(__file__)),
"data.csv")
    return pd.read_csv(data_path, delimiter=";")


def test(title, labels, samples, alternative, discrete=False,
alpha=default_alpha, precision=default_precision):
    all_normally_distributed = check_dist_normality(title, labels, samples,
alpha, precision)
    compare_ind_means(title, labels, samples, all_normally_distributed,
alternative, discrete, alpha, precision)


def check_dist_normality(title, labels, samples, alpha=default_alpha,
precision=default_precision):
    all_normally_distributed = True
    for sample_index, sample in enumerate(samples):
        stat, p = shapiro(sample)
        if p > alpha:
            print(
                f"{title} ({labels[sample_index].lower()}):\n\tH\u2080:
нормальний закон розподілу (p = {p:.{precision}f})",
                end="\n\n")
        else:
            print(
                f"{title} ({labels[sample_index].lower()}):\n\tH\u2081: інший
закон розподілу (p = {p:.{precision}f})",
                end="\n\n")
            all_normally_distributed = False
    return all_normally_distributed


def compare_ind_means(title, labels, samples, all_normally_distributed,
alternative, discrete,
                      alpha=default_alpha, precision=default_precision):
    print(title + " (" + labels[0].lower() + " / " + labels[1].lower() + "):")

    if all_normally_distributed and not discrete:
        variances = tuple(np.var(sample, ddof=1) for sample in samples)
```

```
        dfs = tuple(len(sample) - 1 for sample in samples)
        print(f"s\u2081\u00B2 = {variances[0]:.{precision}f}")
        print(f"s\u2082\u00B2 = {variances[1]:.{precision}f}")

        upper_variance_index, lower_variance_index = (0, 1) if variances[0] >=
variances[1] else (1, 0)
        f_critical_lower = f.ppf(alpha / 2, dfs[upper_variance_index],
dfs[lower_variance_index])
        f_critical_upper = f.ppf(1 - alpha / 2, dfs[upper_variance_index],
dfs[lower_variance_index])
        f_stat = variances[upper_variance_index] /
variances[lower_variance_index]

        if f_critical_lower <= f_stat <= f_critical_upper:
            equal_var = True
            print(f"\tH\u2080: s\u2081\u00B2 = s\u2082\u00B2 (F =
{f_stat:.{precision}f}, F ∉ (-∞, {f_critical_lower:.{precision}f}) ∪
({f_critical_upper:.{precision}f}, +∞))", end="\n\n")
        else:
            equal_var = False
            print(f"\tH\u2081: s\u2081\u00B2 ≠ s\u2082\u00B2 (F =
{f_stat:.{precision}f}, F ∈ (-∞, {f_critical_lower:.{precision}f}) ∪
({f_critical_upper:.{precision}f}, +∞))", end="\n\n")

        ci = t.interval(1 - alpha, df=len(samples[0]) - 1,
loc=np.mean(samples[0]), scale=np.std(samples[0]))
        print(f"x¯\u2081 = {np.mean(samples[0]):.{precision}f}, CI =
({ci[0]:.{precision}f}, {ci[1]:.{precision}f})")

        ci = t.interval(1 - alpha, df=len(samples[1]) - 1,
loc=np.mean(samples[1]), scale=np.std(samples[1]))
        print(f"x¯\u2082 = {np.mean(samples[1]):.{precision}f}, CI =
({ci[0]:.{precision}f}, {ci[1]:.{precision}f})")
        stat, p = ttest_ind(samples[0], samples[1], alternative=alternative,
equal_var=equal_var)

        if alternative == "less":
            zero_sign = "≥"
            alt_sign = "<"
        elif alternative == "greater":
            zero_sign = "≤"
            alt_sign = ">"
        else:
            zero_sign = "="
            alt_sign = "≠"

        if p > alpha:
            print(f"\tH\u2080: x¯\u2081 {zero_sign} x¯\u2082 (p =
{p:.{precision}f})", end="\n\n")
        else:
            print(f"\tH\u2081: x¯\u2081 {alt_sign} x¯\u2082 (p =
{p:.{precision}f})", end="\n\n")
    else:
        print(f"x¯\u2081 = {np.array(samples[0]).mean():.{precision}f}")
        print(f"x¯\u2082 = {np.array(samples[1]).mean():.{precision}f}",
end="\n\n")

        stat, p = mannwhitneyu(samples[0], samples[1])
        print(title + " (" + labels[0].lower() + " / " + labels[1].lower() +
"):")

        if p > alpha:
```

```
            print(f"\tH\u2080: відсутня статистично значуща різниця (p =
{p:.{precision}f})")
        else:
            print(f"\tH\u2081: наявна статистично значуща різниця (p =
{p:.{precision}f})")


def build_box_plots(title, category, value, labels, show_data_points, palette,
samples):
    plt.figure(figsize=(8, 6))

    xs, ys = [], []
    for sample_index, sample in enumerate(samples):
        ys.append(sample)
        xs.append(np.random.normal(sample_index + 1, 0.04, len(sample)))

    plt.boxplot(ys, labels=tuple(label.split()[0] for label in labels),
flierprops=dict(marker=""))

    if show_data_points:
        for x, y, color in zip(xs, ys, palette):
            plt.scatter(x, y, alpha=0.4, color=color)

    median_line = plt.Line2D([], [], color="tab:orange", label="Медіана")
    average_line = None

    means = [np.mean(y) for y in ys]
    for mean_index, mean in enumerate(means):
        average_line = plt.hlines(mean,
                                  mean_index + 1 - 0.2,
                                  mean_index + 1 + 0.2,
                                  color="gray",
                                  linestyle="--",
                                  linewidth=1,
                                  label="Середнє значення")

    plt.title(title, fontweight="bold")
    plt.legend(handles=[median_line, average_line])
    plt.xlabel(category)
    plt.ylabel(value)
    plt.grid(visible=True, axis="y", linestyle=":", linewidth=1)
    plt.subplots_adjust(left=0.1, right=0.95, bottom=0.1, top=0.95)
    plt.show()


def build_likert_scales(title, results, category_names):
    labels = list(results.keys())
    data = np.array(list(results.values()))
    data_cum = data.cumsum(axis=1)
    middle_index = data.shape[1] // 2
    offsets = data[:, range(middle_index)].sum(axis=1) + data[:, middle_index] /
2

    # Карта кольорів
    category_colors = plt.get_cmap("RdYlGn")(np.linspace(0.15, 0.85,
data.shape[1]))

    fig, ax = plt.subplots(figsize=(10, 5))

    row_count = len(results)
    column_count = len(category_names)

    int_sums = [0] * row_count
```

```
    remainders = []
    for row_index in range(row_count):
        for column_index in range(column_count):
            int_sums[row_index] += int(data[row_index][column_index])
        remainders.append(100 - int_sums[row_index])

    rounding = [[]] * row_count
    for row_index in range(row_count):
        for column_index in range(column_count):
            rounding[row_index].append((data[row_index][column_index],
column_index))
        rounding[row_index] = sorted(rounding[row_index], key=lambda x: x[0] %
1, reverse=True)

    # Смуги
    for column_index, (colname, color) in enumerate(zip(category_names,
category_colors)):
        widths = data[:, column_index]
        starts = data_cum[:, column_index] - widths - offsets
        rects = ax.barh(labels, widths, left=starts, height=0.35, label=colname,
color=color)

        for row_index, rect in enumerate(rects):
            width = rect.get_width()
            if width == 0:
                continue
            x_pos = rect.get_x() + width / 2
            y_pos = rect.get_y() + rect.get_height() / 2

            index = next((i for i, tup in enumerate(rounding[row_index]) if
tup[1] == column_index), None)
            if index < remainders[row_index]:
                label_text = f"{math.ceil(width)}%"
            else:
                label_text = f"{int(width)}%"

            ax.text(x_pos, y_pos, label_text, ha="center", va="center",
color="black")

    # Лінія нуля
    ax.axvline(0, linestyle="--", color="black", alpha=0.25)

    # Вісь X
    ax.set_xlim(-100, 100)
    ax.set_xticks(np.arange(-100, 101, 10))
    ax.xaxis.set_major_formatter(lambda x, pos: str(abs(int(x))))

    # Вісь Y
    ax.invert_yaxis()
    ax.tick_params(axis="y", which="major", length=0)

    # Заголовок
    ax.set_title(title, pad=27, fontweight="bold")

    # Легенда
    ax.legend(ncol=column_count,
              bbox_to_anchor=(0.5, 1),
              loc="lower center",
              fontsize="small")

    # Фон
    fig.set_facecolor("#FFFFFF")
    plt.show()
```

```
def build_mean_bars(title, category, value, labels, palette, samples,
precision=default_precision):
    means = tuple(sum(sample) / len(sample) for sample in samples)
    plt.figure(figsize=(8, 6))

    bars = plt.bar(labels, means, color=palette)
    plt.title(title, fontweight="bold")
    plt.xlabel(category)
    plt.ylabel(value)

    for bar, mean in zip(bars, means):
        plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() / 2,
f"{mean:.{precision}f}",
                 ha="center", va="center", fontweight="bold", color="white",
fontsize="x-large")

    plt.show()
```

ДОДАТОК Л

АКТ УПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

ЗАТВЕРДЖУЮ
Перший проректор Сумського державного університету
(посада)
(підпис) Інна ШКОЛЬНИК
(ім'я та ПРІЗВИЩЕ)
"___" _____ 2024 р.

АКТ
впровадження (використання) результатів
кваліфікаційної роботи магістра Сивоконя Вадима Володимировича на тему «Інформаційна технологія проєктування віртуального навчального тренажера на основі скінченного автомата з додатковою пам'яттю» у навчальний процес

Комісія в складі:

Голова комісії: голова ради з якості факультету електроніки та інформаційних технологій, к.ф-м.н., доцент — Ірина ПАЗУХА (ім'я та ПРІЗВИЩЕ)

Члени комісії:

заступник декана факультету електроніки та інформаційних технологій, к.ф.-м.н., доцент (посада) — Тарас ЛЮТИЙ (ім'я та ПРІЗВИЩЕ)

в.о. завідувача кафедри комп'ютерних наук, к.т.н., доцент (посада) — Ігор ШЕЛЕХОВ (ім'я та ПРІЗВИЩЕ)

заступник завідувача кафедри комп'ютерних наук, к.ф.-м.н., доцент (посада) — Оксана ШОВКОПЛЯС (ім'я та ПРІЗВИЩЕ)

Встановила, що результати кваліфікаційної роботи використовуються в навчальному процесі за освітньою програмою «Інформатика» (назва програми)

освітнього ступеня бакалавр (бакалавр, магістр, доктор філософії) спеціальності 122 Комп'ютерні науки (назва спеціальності)

шляхом реалізації наступного: із застосуванням інформаційної технології проєктування віртуального навчального тренажера на основі скінченного автомата з додатковою пам'яттю розроблено інформаційне та програмне забезпечення інтерактивного навчального об'єкта з теми «Транспортні задачі» для використання на заняттях із дисципліни «Математичні методи дослідження операцій».

"13" травня 2024 р.

Голова комісії: (підпис) Ірина ПАЗУХА (ім'я та ПРІЗВИЩЕ)

Члени комісії: (підпис) Тарас ЛЮТИЙ (ім'я та ПРІЗВИЩЕ)

(підпис) Ігор ШЕЛЕХОВ (ім'я та ПРІЗВИЩЕ)

(підпис) Оксана ШОВКОПЛЯС (ім'я та ПРІЗВИЩЕ)