МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Програмна реалізація алгоритмів чисельних методів»

Завідувач випускаючої кафедри — **А. С. Довбиш**

Керівник роботи — **О. А. Шовкопляс**

Студент групи ІН-71 — **А. С. Грицина**

СУМИ 2021

Сумський державний університет

(назва вузу)

Факультет ЕлІТ Кафедра Комп'ютерних наук
Спеціальність «Комп'ютерні науки»

Затверджую_____
Зав. кафедри Довбиш А. С.
«_____» _____20___ р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТУ

Грицині Андрію Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація алгоритмів чисельних методів
затверджую наказом по університету від «____» _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проєкту (роботи) _____

3. Вхідні дані до проєкту (роботи)_____
_____
_____
_____
_____

4. Зміст пояснювальної записки (перелік питань, що їх належить розробити) 1) Інформаційний огляд веб-додатків. Аналіз ринку веб-додатків. Структура веб-додатку. Етапи розробки веб-додатку. Аналіз існуючих рішень. Постановка задачі. 2)Аналітична частина. Технології розробки додатку. Логіка роботи додатку. 3)Практична частина. Інтеграція React у Electron. Створення загального функціоналу.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
_____
_____
_____
_____
_____

6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Термін виконання проєкту (роботи)	Примітка
1.	Аналіз проблеми. Постановка задачі дослідження	05.04–13.04.2021	
2.	Аналіз технологій та інструментів розробки додатку.	14.04–11.04.2021	
3.	Розроблення структури й логіки додатку.	12.04–19.04.2021	
4.	Розроблення програмної реалізації алгоритмів чисельних методів.	20.04–20.05.2021	
5.	Оформлення пояснювальної записки до дипломної роботи	20.05–04.06.2021	

Студент–дипломник _____
(підпис)

Керівник проєкту _____
(підпис)

РЕФЕРАТ

Записка: 56 стор., 23 рис., 1 табл., 1 додаток, 20 джерел.

Об’єкт дослідження – створення додатків за допомогою веб-технологій.

Мета роботи – створення додатку для реалізації алгоритмів чисельних методів**.**

Методи дослідження – розробка додатку за допомогою веб-технологій.

Результати – розроблений додаток, орієнтований на навчання. Додаток надає можливості реалізації алгоритмів чисельних методів та містить повну теоретичну інформацію до вибраного методу. Наявне меню, графічна частина та можливість зміни мови. Розроблений додаток реалізований за допомогою мов програмування та фреймворків: HTML, Javascript, React, Electron.

Зміст

ВСТУП

На сучасному ринку праці програмісти займають топи рейтингів маючи можливість знайти високооплачувану роботу як в Україні, так і поза її межами[1]. Для того щоб відповідати трендам світового IT-ринку фахівець повинен бути висококваліфікованим. Підготовка, яку отримує майбутній спеціаліст під час навчання, залежить від багатьох факторів, проте найбільш виділяється якість математичної освіти. З плином часу навчальні програми інтегрують комп'ютерні технології у навчання, розширюючи можливості студента при отриманні та акумулюванні отриманих знань. Викладач має на меті надати вичерпні знання з предмета, при цьому максимально допомагаючи студентові їх засвоїти. У свою чергу студент використовує наявні інструменти аби оволодіти матеріалом. В обох випадках ефективність використання інформаційних технологій несе з собою ряд переваг:

- використання комп'ютерних програм активізує розумову, мовну та фізичну діяльність людини, сприяючи більш повному засвоєнню матеріалу;
- швидкий контроль і допомога з боку викладача;
- можливість для учня активно брати участь в процесі засвоєння матеріалу;
- різні способи подачі інформації та високий рівень її наглядності;
- загальне скорочення часу для вивчення необхідної теми[2].

Більш конкретно говорячи про економію часу варто зазначити, що доволі часто весь потенціал сучасних технологій не використовується у контексті засвоєння матеріалу. Це виникає у випадку коли теоретична частина теми, що викладається, існує у вигляді посібника з теорією та прикладами розв'язку практичних завдань, але коли студент доходить до пункту самостійного опрацювання матеріалу, то у нього виникає проблема відсутності нових наглядних засобів вирішення. З однієї сторони це спонукає учня займатись дослідницькою роботою, більш детально вивчивши сутність

матеріалу за темою, що безумовно стає перевагою при вирішенні більш поглиблених завдань. З іншої сторони, втрата часу, яку створить нерозуміння студентом матеріалу більш нуж помітна, а беручи до уваги насиченість навчальних програм, викладач не завжди зможе цьому запобігти. З саме такою проблемою я зіткнувся при вивченні матеріалу за темою алгоритмів чисельних методів. Виконуючи програмну реалізацію цих алгоритмів я не мав єдиного чіткого способу перевірити якість засвоєного матеріалу, тому ,як було вказано, це призвело до втрати часу. Дослідивши ці алгоритми прийшов до висновку, що процес вивчення цієї теми можна прискорити, акумулювавши всі отримані знання у вигляді додатку.

Об’єктом дослідження є створення додатків за допомогою веб-технологій. Вивчаються такі особливості як:

- простота використання;
- зрозумілість користувацького інтерфейсу;
- функціональні вимоги;
- відповідність очікуванню користувача.

В процесі дослідження вирішувалися наступні завдання:

- аналіз ринку веб-додатків;
- огляд етапів розробки веб-додатків;
- пошук необхідних технологій розробки.

Метою роботи є створення додатку для реалізації алгоритмів чисельних методів. Наукова новизна дослідження полягає в створенні додатку, який реалізує алгоритми чисельних методів та містить у собі необхідну та докладну теорію за темою. Практична значущість результатів дослідження використовується в процесі навчання майбутніх спеціалістів.

1. ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Загальні відомості про веб-додатки

Веб-додаток - це комп’ютерна програма клієнт-сервер, яка використовує веб-браузери і веб-технології для виконання завдань через Інтернет. У нього є клієнтська і серверна частини. Програма, що запускає людина, є клієнтом. За допомогою клієнт-серверного середовища здійснюється обмін інформацією. Сервер – додаток, де зберігається інформація. . Веб-додатку потрібно веб-сервер для управління запитами від клієнта, сервер додатків для виконання запитаних завдань і, іноді, база даних для зберігання інформації. Все, що потрібно для доступу до веб-додатку, - це підключення до Інтернету. Для підключення до додатку використовується веб-браузер. Розробники кодують веб-додатки двома типами мов. Серверний скрипт займається зберіганням та отриманням інформації та вимагає таких мов, як Python або Java. Розробники програмують на сервері для створення сценаріїв, які буде використовувати веб-програма[3]. Для сценарію на стороні клієнта потрібні такі мови, як JavaScript, каскадні таблиці стилів (CSS) та HTML5. Ці мови покладаються на браузер для запуску програми. Вони підтримуються мовами браузера. Сценарій на стороні клієнта займається поданням інформації користувачеві. Деякі додатки є динамічними і вимагають обробки на стороні сервера, інші повністю статичні і не вимагають обробки на сервері

Веб-додаток працює наступним чином:

1. користувач створює запит до веб-сервера через Інтернет через користувальницький інтерфейс програми;
2. веб-сервер надсилає цей запит серверу веб-додатків;
3. сервер веб-додатків виконує запитане завдання, а потім генерує результати необхідних даних;
4. сервер веб-додатків надсилає ці результати назад веб-серверу;

5. веб-сервер передає запитувану інформацію клієнту;
6. запитана інформація відображається на дисплеї користувача.

Переваги веб-додатків:

- веб-додатки працюють на декількох платформах, незалежно від ОС або пристроїв, якщо браузер сумісний;
- усі користувачі отримують доступ до однієї версії, що видаляє будь-які проблеми сумісності;
- не встановлюються на жорсткому диску, що видаляє обмеження по місту.

Недоліки веб-додатків:

- відсутність підтримки стану сеансу роботи і затримка при перезавантаженні кожної сторінки. Перевантаження викликає помітну затримку, викликану необхідністю встановити HTTP- з'єднання, обробити запит на сервері, передати по мережі у відповідь HTTP-повідомлення і перезавантажити сторінку браузером. Це створює скачки та періодичною очисткою роботи користувача;
- обмежений набір елементів управління;
- неузгоджені підходи до оформлення та способу взаємодії з користувачами;
- відкритість доступу до додатків[4].

1.2 Аналіз ринку веб-додатків

Цифрова економіка та зміни спричинені COVID –визначають значимість інтернет-технологій для здійснення всіх видів віддаленої діяльності. Розвиток бізнес-моделей вплинув на веб-розробку, особливо в напрямку вдосконалення маркетингового просування товарів, послуг і компаній. В таких умовах інтереси суспільства змістились до онлайн-простору, і як наслідок актуалізується задача забезпечення інноваційного зростання за допомогою присутності бізнесу в онлайн-просторі, де і по теперішній час наростають

обсяги професіональної та купівельної поведінки. Більшість агентств вищого і середнього цінового ешелону не стали знижувати ціни на послуги, працюючи в звичайному режимі. У той же час для агентств нижнього цінового сегмента були характерні експерименти зі зниженням цін. Однак більша частина з них не отримала бажаних результатів: якщо замовник не має грошей, то немає сенсу в будь-яких заходах. Представників сегмента преміум цієї весни підняли свої розцінки для нових замовників[5]. Ринок фрілансу, який розвивався швидкими темпами щорічно, маючи на кінець 2018 року приріст виконавців в 45% за рік, на кінець 2019 року - 58%, уповільнився у 2020 році, зменшившись до 36% та досягшого рівня 2017 року. Стрибок у кількості виконавців відбувся завдяки карантину та припав на березень 2020 року, збільшуючись аж до травня. Причинами стали пошук звільненими спеціалістами шляхів заробітку та збільшення кількості звернень підприємств до аутсорсу з метою економії. На Рис. 1.1 можна відслідкувати зміну ринку роботи на фрілансі.

Рис. 1.1. Зміна ринку роботи на фрілансі[6].

Для продуктивної комунікації з клієнтом або замовником необхідно мати надійний та незалежний майданчик, яким виступає корпоративний сайт. На теперішній час такі сайти перестали бути лише місцем організації в Інтернеті, вони виступають інструментом вирішення завдань бізнесу: ділове спілкування співробітників компанії, забезпечення комунікації між групами людей, надання інформації про діяльність, послуги та товари компанії. Таким чином відбувається підтримка функції управління та інформаційної функції. Також сайт є іміджевим інструментом, тому для користувача створюються необхідні умови аби отримати вичерпну інформацію: історію компанії, виконані проєкти, каталоги товарів та послуг, наявна можливість зворотного зв'язку.

Корпоративний сайт має відповідати наступним вимогам:

- інтуїтивно зрозумілий інтерфейс;
- швидкість та стабільність;
- адаптивність[7]..

Клієнт, як правило, звертається до ресурсів соціальних мереж для першого, ознайомлювального контакту, але для виникнення відносин, які сприяють укладанню довгострокових домовленостей компанії необхідна наявність власного онлайн-простору, аналогічно офісу в фізичної реальності. Володіння власною «територією» в кібер-просторі у формі корпоративного сайту дозволяє забезпечити компанії як репутаційні переваги, так і деяку незалежність від волюнтаристських рішень або помилок цензурування соціальних мереж. Також компанії маю бути доступними у соціальних мережах для вирішення щохвилинних питань бізнес-клієнтів Сьогодні значення інтернет-технологій в житті людей зросла настільки, що створення корпоративного сайту в якості веб-майданчика бізнесу є необхідною умовою для розвитку практично будь-якого виду підприємств і організацій будь-якої сфери діяльності, розміру і форми власності. Компанії зацікавлені в розширенні і еволюції, актуалізації своєї представленості в Інтернеті, від

проблем переходу клієнтів на мобільні пристрої на початку-середині 2010-х років, і до сучасних тенденцій в дизайні. Також варто звернути увагу на механіку процесу веб-розробки, яка визначає вибір клієнтом форми вирішення завдань в залежності від обсягу очікуваних послуг в діапазоні від мінімуму до «нескінченності»: власний веб-майстер може ефективно підтримувати сайт або сторінку і може виконати разову роботу по створенню найпростішого сайту.

В основі сучасних сайтів лежать спеціальні системи управління - CMS CMS - «стрижень сайту», програма для управління контентом.

Вона включає наступні функції:

- створення і редагування меню;
- наповнення сторінок інформацією;
- управління текстовими блоками.

Користувач CMS працює в графічному інтерфейсі, йому не потрібно знання мов програмування, розмітки HTML і серверних налаштувань. Великі веб-студії розробляють сайти на власній системі управління, наприклад, CMS.S3. Розробка на якісних CMS відрізняється широким функціоналом, інтуїтивно-зрозумілим інтерфейсом і простотою редагування сторінок, що робить його підтримку більш привабливою для замовника, оскільки далі він може самостійно за допомогою своїх фахівців виконувати найпростіші завдання, наприклад, оновлювати інформацію і вносити нові дані.

Про прогресивні веб-додатки почали говорити ще кілька років тому, і з кожним роком технологія привертає увагу інвесторів і розробників. Instagram, Twitter, Starbucks, Pinterest, Aviasales та інші бренди використовують додатки на базі PWA в якості основного продукту або мобільного додатку.

Переваги PWA в наступному:

- можливість перегляду веб-сайтів в мобільному браузері швидкість та зручність, яких наближена до комп’ютерних;
- скорочений, в порівнянні з нативними додатками, процес розробки;

- єдиний досвід використання продукту з будь-якого девайсу;
- PWA, які є веб-сторінками, розпізнаються пошуковими системами.

Однак разом з цим PWA мають і мінуси:

- при відсутності нативного додатки втрачається трафік з App Store і Google Play;
- функціонал більш обмежений аніж у нативних додатків;
- швидко розряджають батарею девайсу[8].

Edtech – це нова практика на безкоштовній основі, яка використовується в області освіти та полягає у впроваджені технологій та нестандартних рішень, ставлячи метою покращення засвоєння та обміну знань. Очікується, що рішення освітніх технологій будуть розвиватися відповідно до досягнень в новітніх технологіях, таких як Інтернет речей, штучний інтелект , доповнена реальність і віртуальна реальність , а також вносять значний вклад в зростання ринку. Інтеграція AR і VR в рішення EdTech допомагає пропонувати учням інтерактивний досвід. Це дозволяє учням вивчати абстрактні концепції і легко зв'язуватися з ними, що згодом сприяє залученню. З іншого боку, інтеграція технології блокчейн дозволяє кінцевим користувачам зберігати і захищати записи про учнів, тим самим дозволяючи викладачам аналізувати моделі споживання матеріалів, запропонованих учням, і приймати рішення на основі даних. Підвищення залученості студентів стає головним завданням викладачів. Отже, учасники ринку реагують на такі побоювання, впроваджуючи сучасні інтерактивні технології і переходячи від дисплеїв на основі проектора до дисплеїв з сенсорним екраном. Такі ініціативи заохочують активне навчання і розвивають у учнів критичні навички. Крім того, як викладачі, так і учні можуть отримати доступ до інформаційних систем для учнів, основним завданням яких є створення всеосяжних профілів учнів, які можуть дозволити викладачам приймати обґрунтовані рішення з особливим наголосом на покращенні навчання кожного учня. Очікується, що глобальний ринок EdTech буде рости із середньорічними темпами зростання 19,9% з 2021

по 2028 рік і до 2028 року досягне 377,85 млрд доларів США. Освітні установи і приватні педагоги пристосовуються до нових реалій і потребують зручних інструментах для онлайн-навчання. В тому числі, коли вчителі проводять заняття в форматі в реального часу, і коли учні вчаться по заздалегідь записаних лекціях і виконують по ним завдання[9].

1.3 Структура веб-додатку

Додатки в основному складаються з логічних частин, кожна з яких має свою роль. Зазвичай такі частини називають «шарами», які утворюють слої. Найпоширенішою є структура з трьох шарів:

- шар уявлення – складається з UI компонент та логіки уявлення;
- шар бізнес-логіки – містить у собі логіку, сутності та процеси;
- сховище – надає доступ до даних.

При створенні веб-додатків особлива увага приділяється ключовим компонентам:

- архітектура веб-додатків;
- веб-клієнт;
- веб-сервер;
- протоколи зв'язку;
- сховища даних;
- візуальні елементи.

Веб-клієнт - це віддалений комп'ютер веб-системи, що використовує ресурси сервера. Веб - клієнти можуть ділитися на кілька типів за різними характеристиками:

- по використовуваному протоколу передачі даних;
- за типом використовуваного додатка;
- по товщині.

Веб-сервери обробляють клієнтські запити, надсилають їх та формують на основі запитів звернення до зовнішніх джерел даних. Веб-сервер працює з

різними протоколами передачі даних, забезпечуючи клієнт-серверне взаємодію і не призначений для обробки користувальницьких подій без формування клієнтського запиту. Будь-яка дія користувача формується на стороні клієнта у вигляді запиту[10].

Протоколи зв'язку забезпечують взаємодію між клієнтом і сервером. Клієнт посилає запит, використовуючи протокол, обробник якого налаштований на сервері. На даний момент найпоширенішими протоколами, що використовуються в веб-додатках, є:

- HTTP - застосовується при взаємодії веб-браузера з веб-сервером;
- RM1 - створений для організації взаємодії між Java-об'єктами, що знаходяться на різних платформах;
- GIOP - стандартний протокол, заснований на технології CORBA. У завдання протоколу входить організація взаємодії між додатками, написаними на різних мовах[11].

Сховища даних вирішують задачу зберігання інформації в зовнішніх, щодо програми, джерелах. Такими джерелами можуть бути різні бази даних, файлові системи і зовнішні носії інформації, які містять програмні елементи.

Візуальні елементи призначені для відображення інформації в формі, що полегшує розуміння переданої інформації. Найчастіше це різні HTML-елементи, видимі в веб-браузері. При цьому під візуальними елементами мається на увазі не тільки графічний інтерфейс клієнтської частини, а й адміністративна консоль управління додатком на сервері.

Архітектура програмного забезпечення - це скелет та всі компоненти високого рівня системи, включаючи їх взаємодію. Веб-додатки охоплюють два різних набору програм, які працюють окремо, але одночасно із загальною метою гармонійної роботи для надання рішень. Як правило, два набори програм включають код в браузері, який працює відповідно до введення користувача, і код на сервері, який працює відповідно до запитів протоколів. Іншими словами, веб-розробники повинні мати можливість приймати рішення

про функції коду на сервері й функціях коду в браузері, а також про те, як вони будуть працювати по відношенню один до одного.

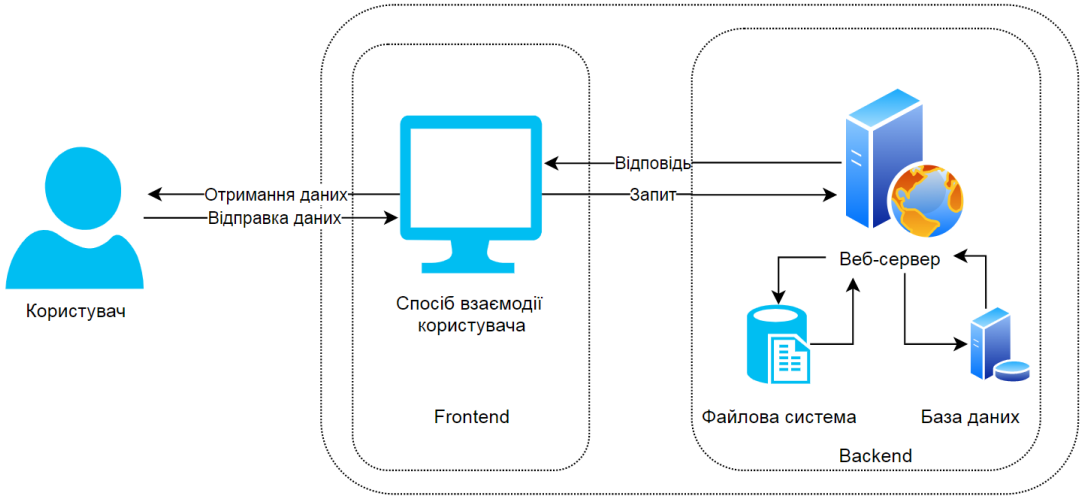


Рис. 1.2. Архітектура веб-додатку

На Рис. 1.2 зображено архітектуру в загальному вигляді. Завдяки програмної архітектурі стає легше побачити загальну картину. Її основна мета - виявити як функціональні вимоги, так і вимоги до якості і обробити їх для поліпшення загальної якості додатку, що в подальшому надає можливість контролювати продуктивність, масштабованість і надійність. Вибір архітектури додатку залежить від мети його створення. . Однорівнева архітектура характеризується відсутністю мережевих сервісів, а зовнішні джерела даних знаходяться практично завжди на одному комп'ютері або в одному сегменті мережі. Дворівнева архітектура відрізняється наявністю веб-сервера та файлової системи на одному комп'ютері, а бази даних на іншому. Типи архітектури веб-додатку включають SPA, мікросервіси та безсерверну архітектуру.

SPA - одна сторінка, яка постійно взаємодіє з користувачем, динамічно переписуючи поточну сторінку. Не завантажує цілі нові сторінки з сервера, надаючи користувачеві більш динамічну взаємодію з веб-додатком. Особливість архітектури SPA полягає в тому, що всі елементи, необхідні для

роботи софта знаходяться на одній сторінці. Вони завантажуються при ініціалізації. Також даний вид додатків завантажує додаткові модулі після запиту від користувача. Будь-яка призначена для користувача активність фіксується для зручності навігації. Архітектурний стиль мікросервісів – підхід який полягає у створенні невеликих сервісів, які комунікують між собою за допомогою легковагих механізмів, та які в об'єднанні складають повноцінний додаток. Існує абсолютний мінімум централізованого управління цими сервісами. Самі по собі ці сервіси можуть бути написані на різних мовах і використовувати різні технології зберігання даних. Безсерверна архітектура має на увазі інфраструктуру, підтримувану сторонніми провайдерами, необхідна функціональність пропонується в формі сервісів, що відповідають за процеси аутентифікації, передачі повідомлень та інших операцій[12]. Таблиця 1.1 відображає переваги та недоліки кожної з архітектур.

Таблиця 1.1. Переваги та недоліки типів архітектур

Тип архітектури	Переваги	Недоліки
SPA	Доступність функціоналу з будь-якого пристрою без технічних проблем; Швидкість та продуктивність при повторному завантаженні даних; Відсутність обмеження об'єму використовуваних даних.	Необхідність постійного інтернет-підключення; За відсутності підтримки у користувача JS елементів додаток не працює; Проблеми з індексацією пошуковими системами всіх модулів додатку.

Продовження Таблиця 1.1

Мікросервіси	Покращена ізоляція дефекту компонентів; Відсутня прив'язаність до технологічного стеку; Легке опрацювання новими користувачами.	Важка розробка та компоновка модулів; Тестування проводиться для кожного сервісу, що збільшує об'єм роботи; Складне монтування додатків.
Безсерверна ахітектура	Відсутність необхідності контролю та керування операційними системами; Відсутність необхідності вибирати розмір серверів, відстежувати або масштабувати їх; Відсутність ризику переплати за зайві ресурси; Відсутність ризику спаду продуктивності через недостатній обсяг ресурсів.	Втрата контролю; Процедура тестування ускладнюється через вплетені в структуру сервіси.

До компонентів архітектури веб-додатків входять:

- DNS - фундаментальна система, яка допомагає шукати доменне ім'я та IP-адрес;
- балансир навантаження- в першу чергу займається горизонтальним масштабуванням. Направляючи вхідні запити на один з декількох серверів, балансир навантаження відправляє відповідь користувачеві;
- сервер веб-додатку- цей компонент обробляє запит користувача і надсилає відповідь в браузер;
- бази даних - надає інструменти для організації, додавання, пошуку, оновлення, видалення та виконання обчислень;
- служба хешування - надає сховище для даних, яке дозволяє зберігати і шукати дані. Кожного разу, коли користувач отримує деяку інформацію з сервера, результати цієї операції потрапляють в кеш, таким чином, майбутні запити повертаються швидше;
- сервіси - створюються у вигляді окремих додатків.;
- сховище - модель онлайн-зберігання і обміну даними через Інтернет. Сховище даних можна використовувати для зберігання безлічі файлів різних типів;
- CDN - займається відправкою файлів HTML, файлів CSS, файлів JavaScript і зображень.

Зв’язок між компонентами зображено на Рис. 1.3.

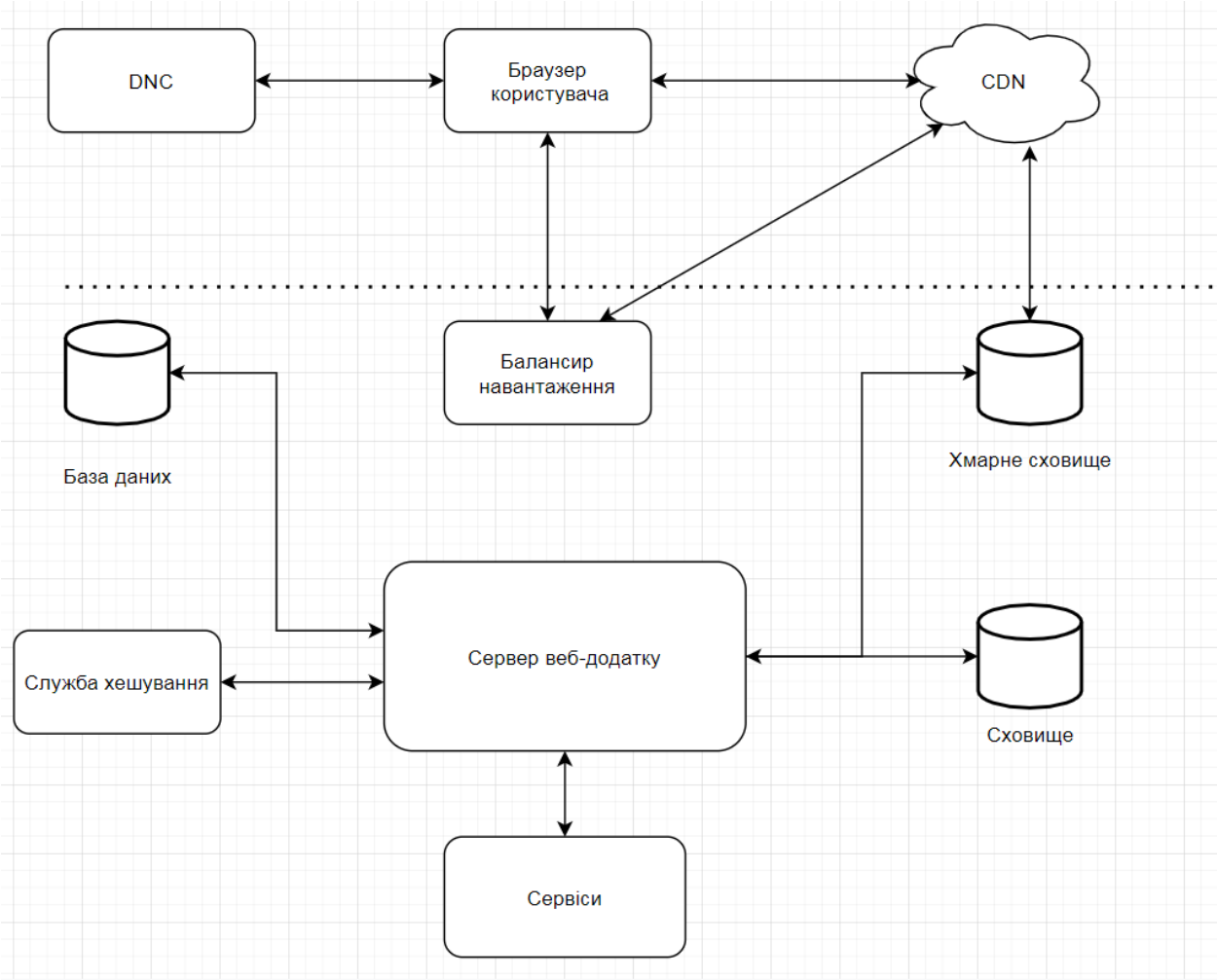


Рис. 1.3. Схема архітектури веб-додатку

1.4 Етапи розробки веб-додатків

Розробка веб-додатку - це комплекс заходів і дій з планування та створення сайту в мережі Інтернет в залежності від поставлених цілей і завдань. Створення технічного завдання є одним з найважливіших етапів розробки сайтів та інших веб додатків. Саме на цьому етапі приймаються всі найважливіші рішення. Для замовника, технічне завдання є тим документом, на який завжди можна послатися, в разі, якщо кінцевий продукт не відповідає поставленому завданню. Якщо виконавець не приділяє достатньо уваги цього етапу, то, з високою ймовірністю, результат його роботи буде далекий від того, що очікував отримати замовник. Винятком можуть бути нескладні проекти, наприклад, прості сайти або сайти-візитки. Для таких проектів технічне

завдання є лише формальністю. Навіть для деяких сайтів середньої складності технічне завдання не завжди обов'язково.

Розробка структури додатку включає все, що стосується його змісту та інформаційної стратегії, яка визначає, як повинна бути організована подача інформації, задля прискорення та полегшення її пошуку майбутніми користувачами. Створюється карта сайту, на якій наявні взаємозв'язки типових сторінок та функціональні можливості.

Дизайн, безсумнівно, один з найважливіших етапів розробки сайтів і веб додатків. Нерідко бувають випадки, коли вирішальну роль грає упаковка, а не якість самого продукту. Саме оформлення задає відношення користувача до додатку. Потрібно пам’ятати, що дизайн- це поняття суб'єктивне, і залежить від індивідуальних особливостей кожної людини. Те, що подобається одній групі людей, може відштовхнути інших. Вкрай бажано мати уявлення про моду і смаки цільової аудиторії. Крім зовнішнього сприйняття, важливим фактором є зручність використання додатку. Розробка інтерфейсу тісно пов'язана з дизайном проекту і тому часто їх розглядають разом, як один етап. Розташування і зовнішній вигляд елементів навігації, інтуїтивне сприйняття цих елементів користувачем, мінімізація дій користувача, - це ті завдання, які доводиться вирішувати дизайнеру при створенні дизайну і навігації. Досить важливим етапом розробки веб додатків, є верстка. Якість виконання цього етапу буде впливати на ряд параметрів додатку, в першу чергу, кросбраузерність, - здатність сайту в різних браузерах відображатися однаково. Верстка також впливає на швидкість завантаження веб сторінки, і, звичайно на стабільність її роботи. Не мало важливим є вплив верстки на обробку сайту пошуковими машинами. Грамотна верстка виконується в суворій відповідності зі специфікацією мови розмітки. Специфікація вказує браузеру, як інтерпретувати переглядаєму сторінку. Програмування є одним з основних етапів розробки веб додатків і сайтів. Найважливішим завданням програміста є вибір найбільш оптимального рішення для досягнення

поставленої мети. Саме тому, бажано заздалегідь, в деталях, продумати весь проект. Інакше, надалі, доробка програми може погано позначитися на її продуктивності. Чим складніше проект, тим важливіше роль програмування в його реалізації, і тим вище рівень кваліфікації потрібно для його виконавця. Етап програмування сайтів і веб додатків можна розбити на різні етапи, - це проектування баз даних, програмування інтерфейсів і програмування основної частини. Одним з найпростіших етапів розробки сайтів і веб додатків є інсталяція. Вся складність цього етапу, як правило, впирається в те, щоб знайти спільну мову з тих. підтримкою сервера. Друга складність в тому, що рідко знаєш заздалегідь налаштування серверу і у випадку, коли щось не працює, потрібно витратити час, щоб дослідити проблему, зрозуміти з ходу причину несправності виходить не завжди. Завершальним етапом є тестування додатку. Задача цього етапу проста – шукати помилки, які виникли при написанні коду. Метою комплексного тестування є перевірка того, що кожен модуль програмного продукту коректно узгоджується з іншими модулями продукту. При комплексному тестуванні може використовуватися технологія обробки зверху вниз і знизу вгору, при якій кожен модуль, який є листом в дереві системи, інтегрується з наступним модулем нижчого або вищого рівня, поки не буде створено дерево програмного продукту. Ця технологія тестування спрямована на перевірку не тільки тих параметрів, які передаються між двома компонентами, а й на перевірку глобальних параметрів і, в разі об'єктно-орієнтованого додатки, всіх класів верхнього рівня.

1.5 Аналіз існуючих рішень

Загалом, вже існуючи рішення представляють в більшості випадків як онлайн-калькулятори. Більшість з них пропонує лише поля для вводу та результат(Рис. 1.4).

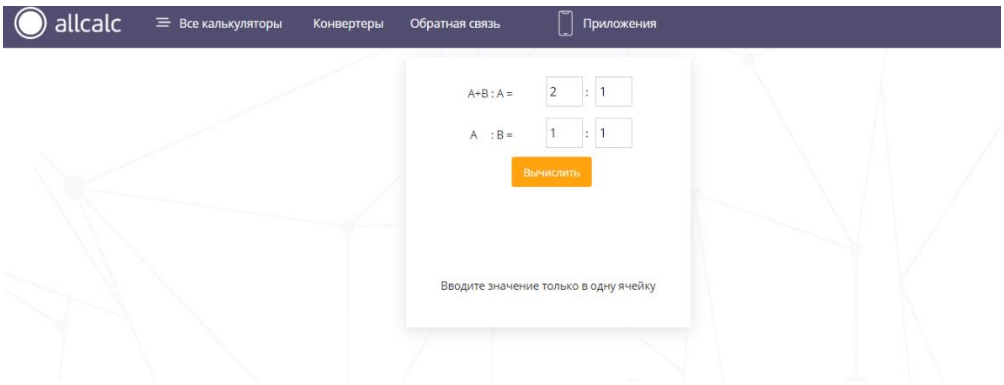


Рис. 1.4. Сторінка онлайн-калькулятора allcalc.ru

Через це зосереджу увагу на тих сервісах, які пропонують розширений функціонал.

Онлайн-калькулятори на сайті planetcalc.ru попри основної частини з полями та результатом(Рис. 1.5), на відміну від більшості, ознайомлюють користувача з основами обраного методу(Рис. 1.6).

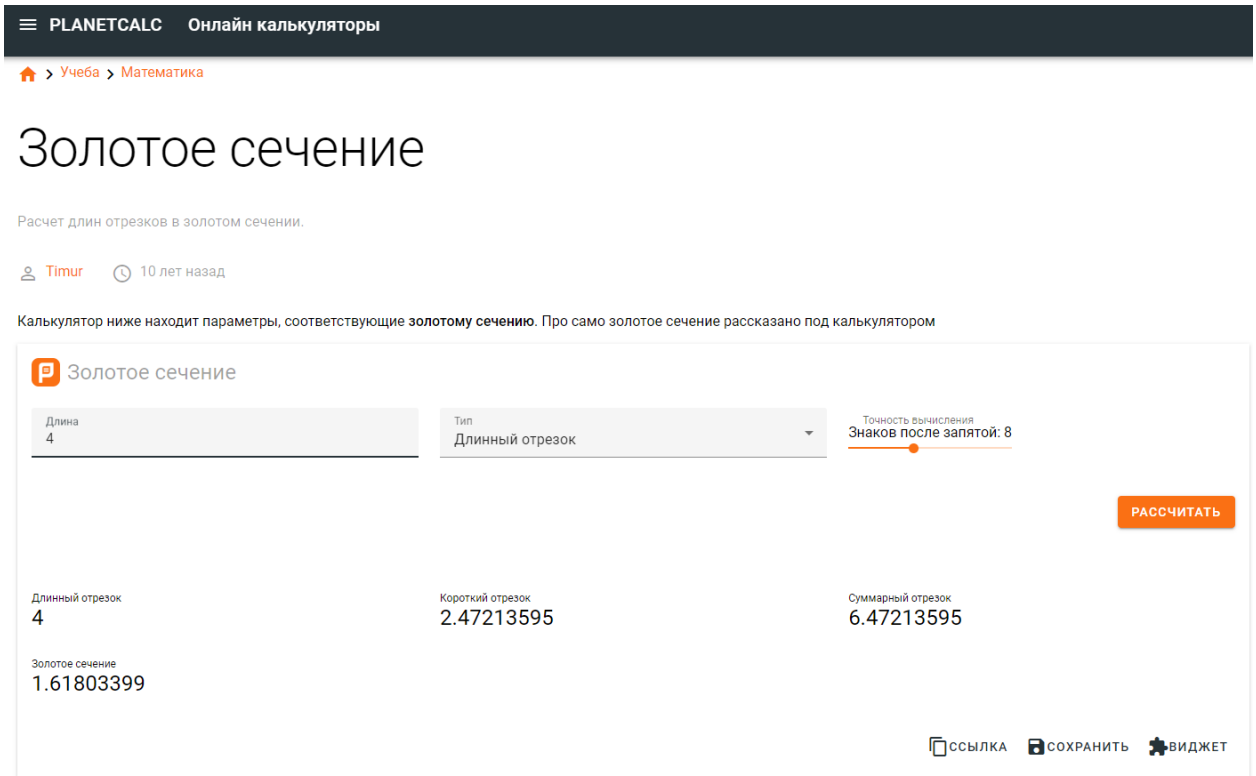


Рис. 1.5. Основна частина сервісу planetcalc.ru

Золотое сечение — термин, обозначающий деление отрезка на два в соотношении, при котором большая часть относится к меньшей также как весь отрезок относится к большей. Также употребляют термин **деление в крайнем и среднем отношении**.

$$\frac{a}{b} = \frac{a+b}{a}$$

Отношение это фиксированное, его можно найти. Представим, что **b** у нас единица. Тогда значение **a** должно равняться искомому отношению, и его надо найти — переименуем его в более привычное **x** и проведем ряд преобразований:

$$\frac{x}{1} = \frac{x+1}{x}$$
$$x = \frac{x+1}{x}$$
$$x^2 = x + 1$$
$$x^2 - x - 1 = 0$$

Последнее есть квадратное уравнение. Его положительный корень:

$$\frac{1+\sqrt{5}}{2}$$

и есть отношение золотого сечения. Число это иррациональное:

$$\frac{\sqrt{5}+1}{2} = 1,6180339887...$$

Для практических целей иногда используют приближение — большая часть равна 0,62 всей величины, меньшая — 0,38 (это видно, если ввести длину 1, и выбрать тип «суммарный отрезок» в калькуляторе сверху).

Золотое сечение известно еще со времен Евклида (ок. 300 лет до н. э.), и у него много забавных свойств, про которые можно почитать в: Википедии, например, к нему стремится отношение последовательных чисел Фибоначчи.

Для полноты ликбеза скажем, что почему-то считается, что объекты, содержащие золотое сечение, воспринимаются людьми как наиболее гармоничные. Ну а вот целая занятная статья, где золотое сечение находят буквально во всем.

Рис. 1.6. Додаткова частина сервісу planetcalc.ru

Наявність теорії це перевага, проте вона загальна, відсутність покрокового вирішення завдання не сприяє розумінню прочитаного матеріалу.

Інший відомий сервіс - math.semestr.ru. Він пропонує як і загальну частину(Рис. 1.7), так і покрокове рішення(Рис. 1.8).

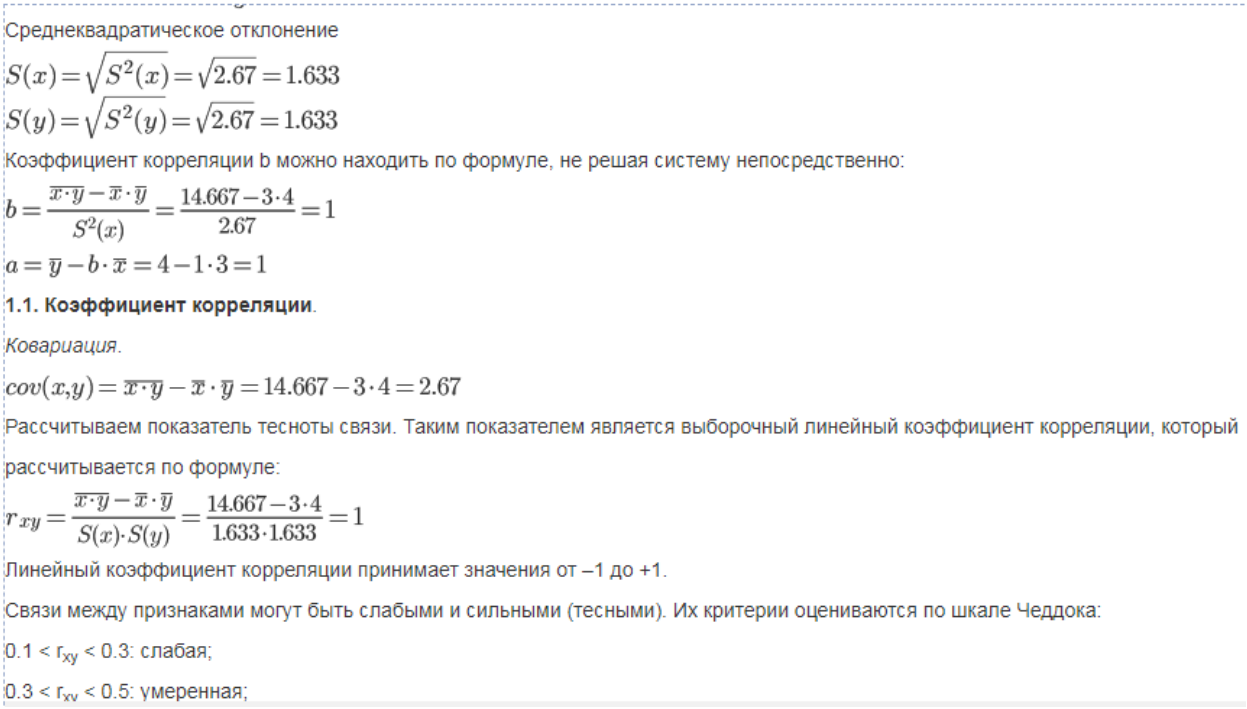
При количестве данных больше 100, создается только шаблон решения в **MS Excel**.

Для получения **обратного уравнения регрессии** *x=by+a* , достаточно вставить данные в обратном порядке (первый столбец *Y*, второй столбец *X*).

X	Y
1	2
3	4
5	6

Итого: n=3

Рис. 1.7. Основна частина сервісу math.semestr.ru



Среднеквадратическое отклонение

$$S(x)=\sqrt{S^2(x)}=\sqrt{2.67}=1.633$$

$$S(y)=\sqrt{S^2(y)}=\sqrt{2.67}=1.633$$

Коэффициент корреляции b можно находить по формуле, не решая систему непосредственно:

$$b=\frac{\overline{x\cdot y}-\overline{x}\cdot\overline{y}}{S^2(x)}=\frac{14.667-3\cdot4}{2.67}=1$$

$$a=\overline{y}-b\cdot\overline{x}=4-1\cdot3=1$$

1.1. Коэффициент корреляции.

Ковариация.

$$cov(x,y)=\overline{x\cdot y}-\overline{x}\cdot\overline{y}=14.667-3\cdot4=2.67$$

Рассчитываем показатель тесноты связи. Таким показателем является выборочный линейный коэффициент корреляции, который рассчитывается по формуле:

$$r_{xy}=\frac{\overline{x\cdot y}-\overline{x}\cdot\overline{y}}{S(x)\cdot S(y)}=\frac{14.667-3\cdot4}{1.633\cdot1.633}=1$$

Линейный коэффициент корреляции принимает значения от –1 до +1.

Связи между признаками могут быть слабыми и сильными (тесными). Их критерии оцениваются по шкале Чеддока:

0.1 < r_{xy} < 0.3: слабая;

0.3 < r_{xy} < 0.5: умеренная;

Рис. 1.8. Додаткова частина сервісу math.semestr.ru

Тут проблема повністю протилежна: скориставшись таким калькулятором користувач нічого не дізнається про теоретичну частину, але зможе відслідкувати процес виконання. Маючи теорію з інших джерел, на яку можна спиратися, використання цього сервісу принесе найбіліший профіт.

Проаналізувавши аналоги, стають помітні їхні переваги та недоліки. При створенні додатку слід перейняти такі корисні особливості як наявність теоретичного матеріалу та покрокового рішення заданої задачі.

1.6 Постановка задачі

Розробити додаток, використовуючи технології розробки веб-додатків, який складається з частини, на якій будуть наявні елементи для взаємодії з користувачем, та обробника, що буде обробляти введені користувачем дані та відтворювати алгоритми чисельних методів, відповідно до запиту користувача. Створити спеціальний підрозділ з всією необхідною теорією за темою.

UI частина повинна містити наступні елементи:

- головна сторінка для відображення наявних сторінок;
- меню для переходу до необхідних алгоритмів;
- поля для вводу формули та початкових даних;
- за можливості відображення - графічна частина.

Обробник виконує наступні задачі:

- реалізація алгоритму;
- попередження про невірно вказані початкові дані;
- запобігання помилок у випадку обробки некоректних даних користувача.

Розділ, що містить теоретичні відомості, повинен вміщати в себе інформацію, яка повністю розкриває суть алгоритму, покроково пояснюючи принцип його роботи. Створення відбувається поетапно відповідно до структури додатку.

2 АНАЛІТИЧНА ЧАСТИНА

2.1 Технології розробки додатку

Для розробки UI частини доцільно використовувати сучасні технології, спираючись на основи, такі як:

- HTML - мова гіпертекстової розмітки, використовується для визначення структури і опису змісту веб-сторінки в структурованій формі;
- CSS - каскадні таблиці стилів, використовуються для опису зовнішнього вигляду веб-контенту;
- JavaScript - мова програмування, використовується для реалізації взаємодії користувача з веб-сайтами та додатками.
- XPath - дозволяє вам вибрати DOM-вузол в документі
- SVG – маштабована векторна графіка, дозволяє описати зображення у вигляді ліній, кривих та інших геометричних фігур.

Для розробки використана відкрита JavaScript бібліотека– React. Вибір здійснено відповідно до результатів опитування State of Frontend, який показав, що понад 43% респондентів використовують React[13]. React це бібліотека для створення користувацьких інтерфейсів. Однією з її характерних особливостей є можливість використовувати JSX. JSX - розширення мови JavaScript. Завдяки ньому React поясную як повинен виглядати UI. JSX виробляє елементи React[14]. React спроектований таким чином, що його впровадження відбувається поступово, використовувати тільки ту функціональність React, яка необхідна в даний момент. React надає можливість створювати ізоморфні додатки. Ізоморфний додаток на React - це односторінкове додаток, яке рендериться на сервері. Сенс серверного рендеринга в тому, що для клієнтів, які не підтримують або обмежено підтримують JavaScript віддаються статичні файли типу .html. Крім того,

ізоморфність React підвищує швидкість завантаження програм і дає користувачам можливість бачити інформацію на сторінці швидше в міру того, як відбувається її завантаження. Швидке завантаження поліпшує користувацький досвід: На сьогоднішній день поведінкові чинники - це одні з найважливіших факторів ранжирування сайту пошукачем Google, важливіше традиційних, наприклад, таких як щільність ключових слів. React - рендерить компоненти сайту як на сервері, так і на стороні клієнта. React добре підходить для створення ізоморфних додатків, так як дозволяє перевикористати майже весь клієнтський код для рендерингу на сервері, в залежності від масштабу додатку. У неізморфних додатках це неможливо через архітектуру. Оскільки, якщо клієнтська частина такого додатка написана на відмінній від серверної частини мови програмування, відсутня можливість перевикористати код, написаний для сервера на клієнті, і навпаки. У випадку з JavaScript, клієнтська частина коду покладається на DOM браузер, якого немає на серверній стороні. React надає абстракцію браузерного DOM у вигляді віртуального. Це дає дві основні переваги:

- код, який працює з віртуальним DOM не залежить від браузера і може виконуватися на сервері;
- оптимізація операції над документами і зниження кількість звернень до браузерних DOM і за рахунок цього значне прискорення роботи Frontend.

Розробка в React стає процесом поліпшення за рахунок таких особливостей:

- компонентно-орієнтований підхід;
- можливість з легкістю змінювати наявні компоненти;
- перевикористання коду.

Замість того, щоб штучно розділити технології, поміщаючи розмітку і логіку в різні файли, React розділяє відповідальність за допомогою слабо пов'язаних одиниць, званих «компоненти», які містять і розмітку, і логіку

Компонент - це самостійна частина інтерфейсу, яка інкапсулює логіку цієї частини, а також описує поведінки і відображення компоненти при різних станах. Завдяки особливості бібліотеки -декларативності, розробник створює опис станів компонент. Декларативний підхід скорочує код і робить його зрозумілим. Компоненти повинні бути перевикористані в різних місцях додатку. Завершений інтерфейс веб-додатку являє собою об’єднання компонентів. Компоненти можуть посилатися на інші компоненти в поверненому ними дереві. Це дозволяє на будь-якому рівні нашого застосування використовувати одну і ту ж абстракцію. Компоненти, які були створені під час роботи над тим чи іншим проектом, не мають додаткових залежностей. Створені компоненти можуть бути з легкістю змінені і використані заново в нових проектах. Взаємодіяючи з елементами, розміщених на формі, представленої у компоненті, користувач здійснює виклик подій, обробники яких створює програміст[15]. Попри стандартні обробники, присутні в JavaScript, React використовує стан. Стан - це інструмент, що дозволяє оновлювати призначений для користувача інтерфейс, грунтуючись на події. При першому відображенні сторінки відбувається рендер, під час якого створюються всі елементи у компоненті, та задається початковий її стан, в якому створюються необхідні змінні. При заданні виконуються прості операції присвоєння, обробка цих даних в стані неможлива. Така конструкція дає можливість динамічно змінювати властивості елементів: взаємодія з певним елементом користувач викликає обробник події, який змінює стан змінної, відповідної за опис іншого елемента. При зміні стану він не змінюється миттєво. React перевіряє, чи не потрібно внести ще якісь зміни, і тільки потім виробляє зміну стану. Це означає, що не можна точно знати, яким буде стан компонента після виклику зміни стану. При зміні стану, ґрунтуючись на попередньому стані, потрібно передавати до стану функцію, а не об'єкт. Ця функція приймає старий стан як

аргумент і повертає об'єкт, який представляє новий стан. Для взаємодії та передачі даних між різними компонентами використовуються пропси.

Пропси представляють колекцію значень, які асоційовані з компонентом. Ці значення дозволяють створювати динамічні компоненти, що не залежать від жорстко закодованих статичних даних. Коли React бачить елемент, який представляє користувальницький компонент, він передає JSX-атрибути цього компоненту у вигляді єдиного об'єкта – пропсів. Додаток з багатьма компонентами вимагає звільнення ресурсів після знищення компонентів. Для цього існують методи життєвого циклу: componentDidMount(), componentWillUnount(). В них можна виконувати логіку, яка виконується після рендеру та при знищенні компоненти.

Життєвий цикл компоненти виглядає наступним чином:

- constructor (props) - конструктор, в якому відбувається початкова ініціалізація компонента;
- static getDerivedStateFromProps (props, state) - викликається безпосередньо перед рендером компонента. Цей метод не має доступу до поточного об'єкту компонента і повинен повертати об'єкт для поновлення об'єкта state або значення null, якщо нічого оновлювати.
- render () - рендеринг компонента;
- componentDidMount () - викликається після рендеру компонента. Тут можна виконувати запити до віддалених ресурсів;
- componentWillUnmount () - викликається перед видаленням компонента з DOM.

Для виконання логіки безпосередньо в створеній компоненті використовуються хуки. Хуки - це функції, за допомогою яких можна «підчепитися» до стану і методів життєвого циклу React з функціональних компонентів. Хуки не працюють всередині класів - вони надають можливість використовувати React без класів.

При використанні хуків потрібно пам’ятати що це JavaScript функції, тому на них діють два обмеження:

- хуки не слід викликати в циклах, умовах або вкладених функцій, лише на верхньому рівні
- хуки не слід викликати із звичайних функцій JavaScript(за винятком хуків, створених користувачем), виклик має відбуватись із функціональних компонентів React[16].

Для реалізації завдання можна використовувати засоби, які пропонує React, налаштувавши обробник даних користувача всередині структури веб-частини, не використовуючи або імітуючи серверну. Таким чином швидкість обробки інформації незначно впаде, натомість відпадуть всі можливі проблеми серверної частини.

Іншим використаним фреймворком є Electron. Electron це фреймворк для розробки десктопних додатків з використанням HTML, CSS і JavaScript. Вбудовуючи Chromium та Node.js у, Electron дозволяє підтримувати одну кодову базу JavaScript та створювати крос-платформні програми, які працюють у Windows, macOS та Linux без власного досвіду розробки нативних додатків. Використання Electron дозволяє привести веб-додаток до вигляду десктоп– додатку, зі збереженням всіх його особливостей. Переваги використання Electron:

- використання CSS, HTML, JavaScript – у зв’язку з React дозволяє за допомогою базових технологій створити повноцінний додаток;
- використання Chromium для відображення призначеного для користувача інтерфейсу - з Chromium можна отримати доступ до всіх функцій вбудовування хрому. Ще одна особливість полягає в тому, що додаток не вимагає перезапуску навіть після внесення змін до коду;
- є альтернативою PWA – в разі використання Node.js зберігається свобода розробки[17].

Electron знає два різних типи процесів: основний процес і процес рендерингу. При запуску програми Electron запускається основний процес. В одному екземплярі програми завжди є тільки один головний процес. Цей процес не має доступу до API-інтерфейсів DOM , не має вікон і поводиться як процес Node.js Як тільки основний процес ініціалізується, він може відкрити вікна. Ці вікна працюють у своєму власному процесі і називаються процесами візуалізації. У Electron кожна веб-сторінка працює по-своєму. Різниця важлива: процеси рендерингу мають звичний DOM з об'єктами вікна та документа, можуть створювати та відтворювати елементи HTML, а також мають доступні інструменти розробника Chromium, тоді як основний процес - насправді лише процес Node.js. У той же час, доступ до багатьох API надається лише з основного процесу: виклик методів, які виконують операції власного графічного інтерфейсу з процесу візуалізації, є ризикованим, та може призвести до витоку ресурсів, а отже, заборонений. Основний процес управляє та полегшує взаємодію з кожним процесом візуалізації. Процеси візуалізації по черзі взаємодіють з основним процесом, але зазвичай не турбуються про взіємодію з іншими процесами візуалізації. Цьому сприяє багатопроцесорна архітектура Chromium, що дозволяє розробникам передавати об'єкти між процесами і навіть викликати методи одного процесу з іншого. Навіть незважаючи на те, що Electron встановлює суворі правила щодо того, який API можна викликати з якого процесу, це також полегшує зв’язок між процесами. Різниця між основним процесом та приєднаними до нього процесами візуалізації є важливою: кожен із них має різні можливості та відповідає за різні аспекти життєвого циклу вашої програми. Electron поєднує Node.js та Chromium, тобто надаючи доступ до використання усіх API Node, включаючи будь-який npm модуль. Звичайно, це охоплює власні модулі, такі як sqlite або nodegit. Надання доступності Node.js всередині веб-контексту має два важливих наслідки. По-перше, розблокування повного системного доступу до веб-сторінки справді знімає всі обмеження, з якими стикався розробник. З

точки зору операційної системи, додаток Electron має повний доступ до простору користувача та має обмеження доступу, рівна нативному додатку. Він також усуває будь-які межі безпеки, до яких може бути використаний розробник. Скомпрометована програма Electron набагато небезпечніша, ніж скомпрометована веб-сторінка, враховуючи те, що програма Electron має повний доступ до операційної системи. Щоб пом'якшити цей ризик, Electron дозволяє створювати процеси візуалізації, в яких інтеграція Node.js повністю вимкнена[18].

Для задання зовнішнього вигляду сторінки використано мову Sass. Sass - це метамова на основі CSS, призначена для збільшення рівня абстракції CSS-коду і спрощення файлів каскадних таблиць стилів. Використання Sass валіде коли таблиця стилів стає досить великою, і виникає проблема її обслугу. В такому випадку допомогає препроцесор. Sass дозволяє використовувати функції недоступні в самому CSS,

- вкладеність - SCSS дозволяє вкладати правила CSS один в одного. Вкладені правила застосовуються тільки для елементів, відповідних зовнішнім селекторам;
- змінні - в стандартному CSS теж є поняття змінних, але в Sass з ними можна працювати по-іншому. Наприклад, повторювати їх через директиву @for або генерувати властивості динамічно:
- покращені математичні операції-можна додавати, віднімати, множити і ділити значення CSS. На відміну від стандартного CSS, Sass надає можливість обійтися без функції calc ();
- тригонометрія - SCSS дозволяє писати власні функції, використовуючи тільки синтаксис Sass / SCSS, подібно до того, як це можна робити в інших мовах;
- директиви @for, @while і вираз @ if-else - можна писати CSS-код, використовуючи елементи з інших мов, зберігаючи на виході CSS файл;

- домішки - можна один раз створити набір CSS-властивостей і працювати з ними повторно або змішувати з іншими значеннями. Домішки можна використовувати для створення окремих тем одного макета. Домішки також можуть містити цілі CSS-правила або інше, дозволене в Sass-документі, можуть приймати аргументи, що дозволяє створювати велику різноманітність стилів за допомогою невеликої кількості домішок.
- функції - можна створювати визначення CSS у вигляді функцій для багаторазового використання.

При початку користування Sass, препроцесор обробляє Sass-файл і зберігає його як простий CSS-файл, який можна використати на будь-якому сайті[19].

Важливим також є підтримка різних мов. Для цього використано інтернаціоналізацію. Інтернаціоналізація (i18n) - це процес підготовки програмного забезпечення для підтримки місцевих мов і культурних особливостей. Інтернаціоналізований продукт відповідає вимогам місцевих ринків по всьому світу, функціонує більш адекватно на основі місцевих норм і краще відповідає очікуванням користувачів всередині країни. I18n часто помилково приймають за локалізацію (L10n), а іноді навіть за переклад. I18n орієнтований на розробку продукту, тому в разі програмного забезпечення одна кодова база може підтримувати всесвітні мови, а також форматування і поведінку, що залежать від локалі.. Локалізація робить продукт специфічним для цільового ринку або регіону, включаючи переклад інтерфейсу, можливу адаптацію термінології і багато іншого. Типові завдання інтернаціоналізації програмного забезпечення: Розробка програмного забезпечення для незалежності від конкретної мови або обмежувального набору символів та незалежності від культурних традицій

- локальні рамки;
- досягнення відповідності Unicode;

- виключення жорстко закодованого тексту у коді;
- видалення та переробка об'єднаних рядків;
- підтримка збору даних;
- підтримка двонаправлених мов, таких як іврит та арабська[20].

2.2 Логіка роботи додатку

Додаток надає користувачу можливість введення даних, їх обробки та надання результату. Так як підтримується робота декількох алгоритмів, перш за все користувачу буде пропоновано вибрати один із потрібних йому алгоритмів. Зробивши вибір, користувачу буде надано доступ до внесення даних, необхідних додатку для реалізації вибраного алгоритму. Ввівши дані, починається обробка інформації шляхом відтворення обраного алгоритму. Отримавши результат виконання методу користувачу надається очікуваний результат. На основі цієї логіки створено діаграму прецедентів(Рис. 2.1), DFD 0(Рис. 2.2), DFD 1(Рис. 2.3), DFD 2(Рис. 2.4) діаграми та діаграму активностей(Рис. 2.5).

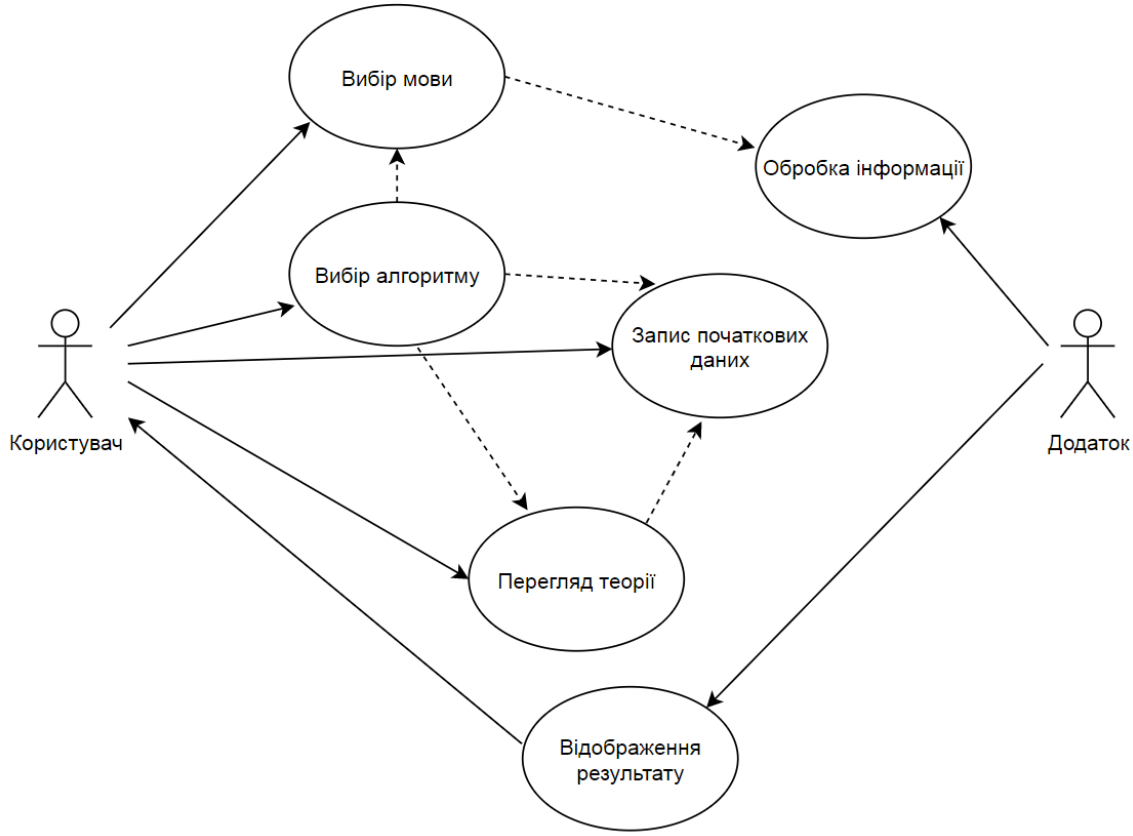


Рис. 2.1. Use-case діаграма

Рис. 2.2. DFD 0 діаграма

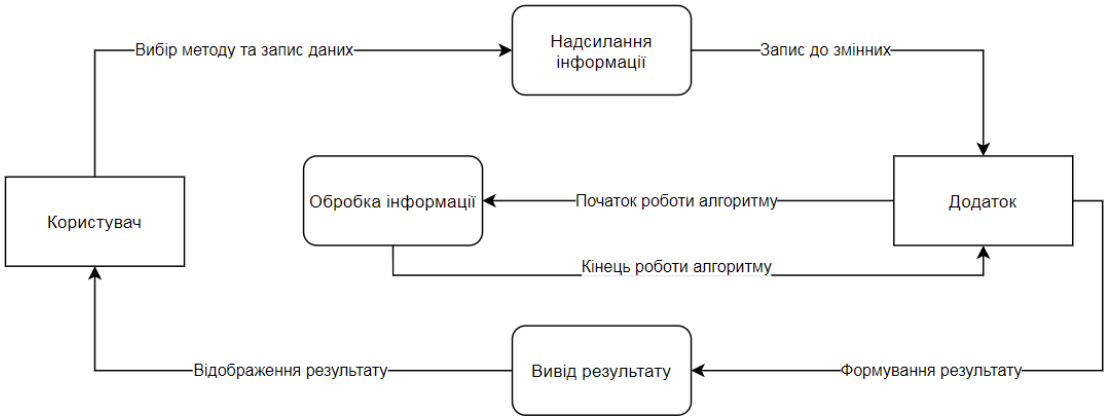


Рис. 2.3. DFD 1 діаграма

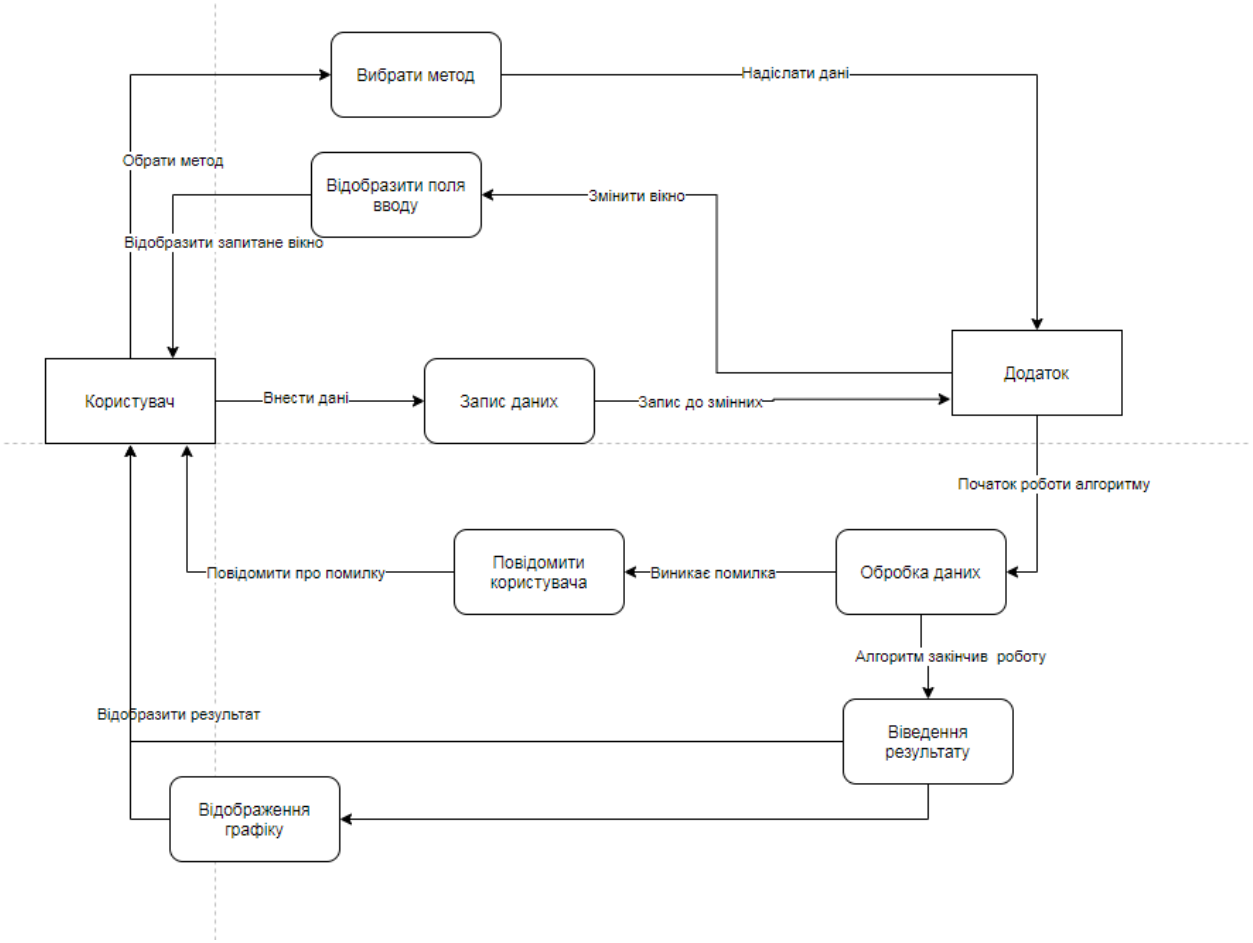


Рис. 2.4. DFD 2 діаграма

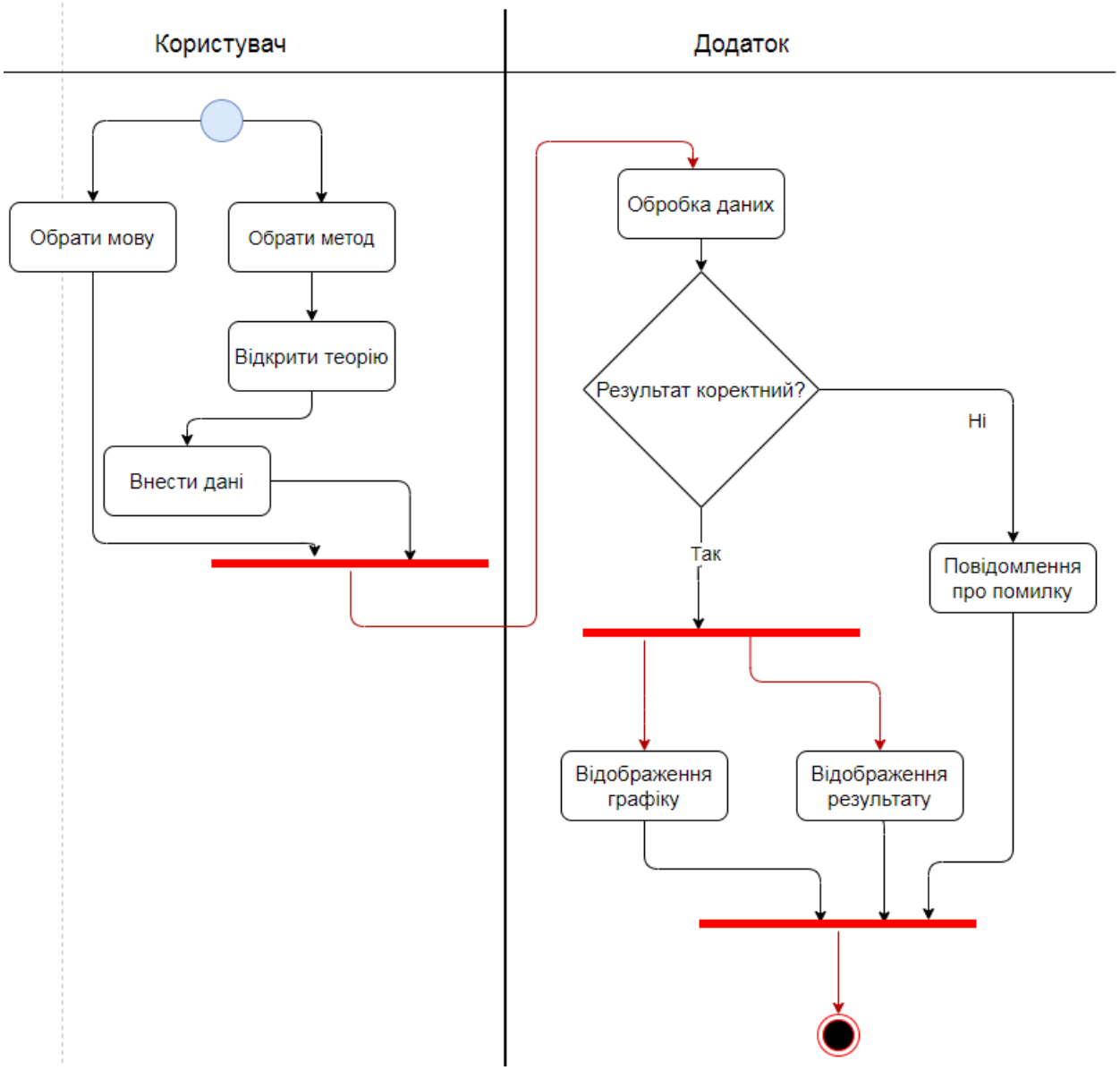


Рис. 2.5. Activity diagram

3 ПРАКТИЧНА ЧАСТИНА

3.1 Інтеграція React у Electron

Для початку створимо шаблон проєкту на React. Для цього відкриємо редактор коду – Visual Studio Code. В ньому відкриємо новостворену папку та створимо термінал(Рис. 3.1).

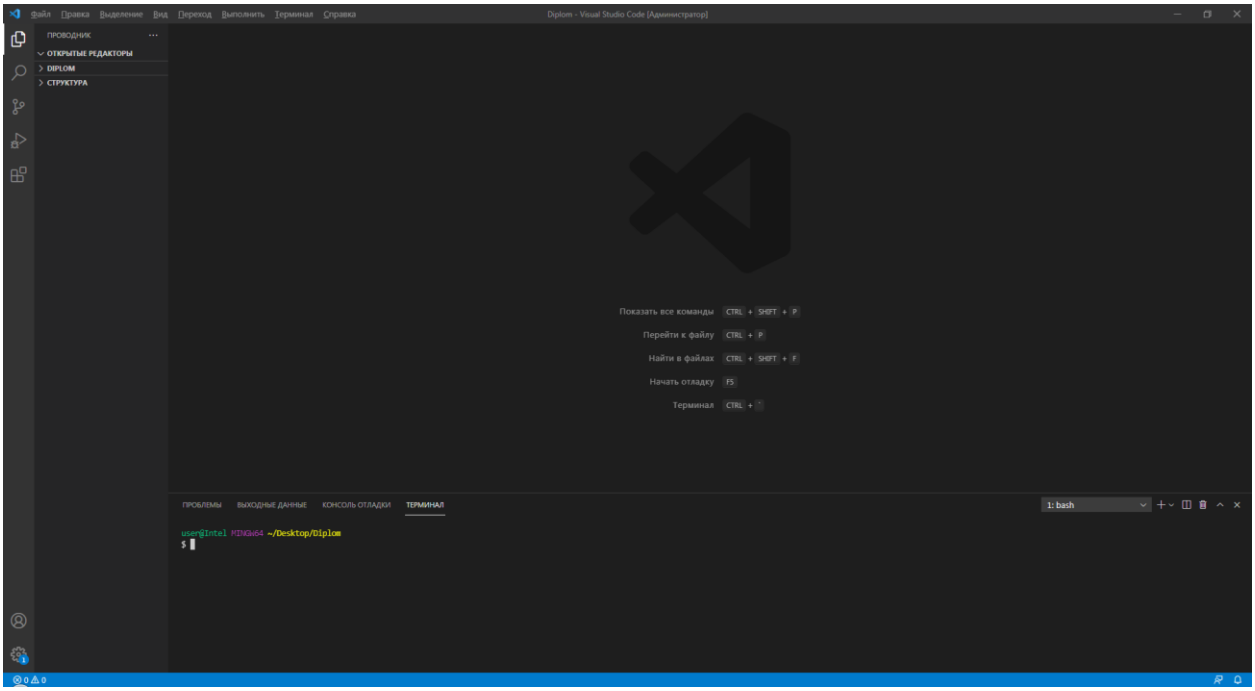


Рис. 3.1. Початок роботи в середовищі

Прописуємо npx create-react-app Diplom. В результаті буде створено шаблон проєкту(Рис. 3.2). В ньому нас цікавлять три файли – index.html, index.js та App.js. Точка входу – index.html відповідає за запуск програми, в ньому викликається index.js який запускає App.js, в якому і відбувається рендер сторінки. Для запуску програми потрібно перейти через термінал в папку проєктута ввести команду npm install, а потім npm start. Відкриється вікно браузера за замовченням з локальним портом 3000(Рис. 3.3).

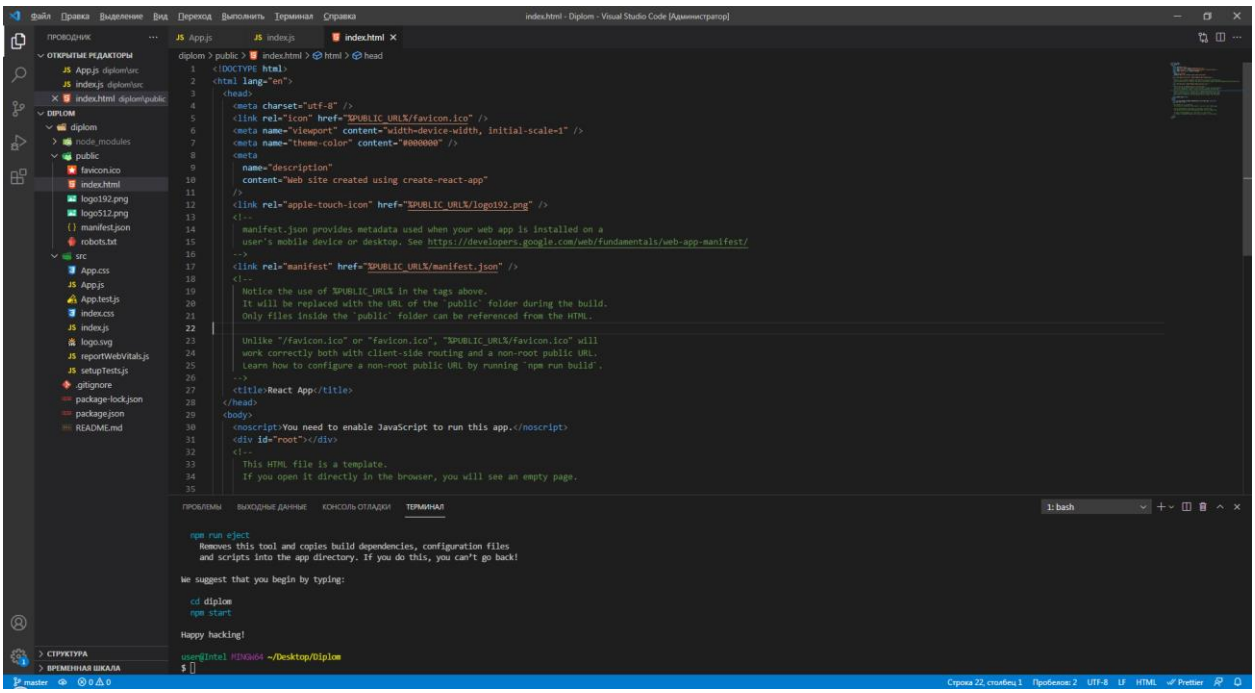


Рис. 3.2. Створений шаблон

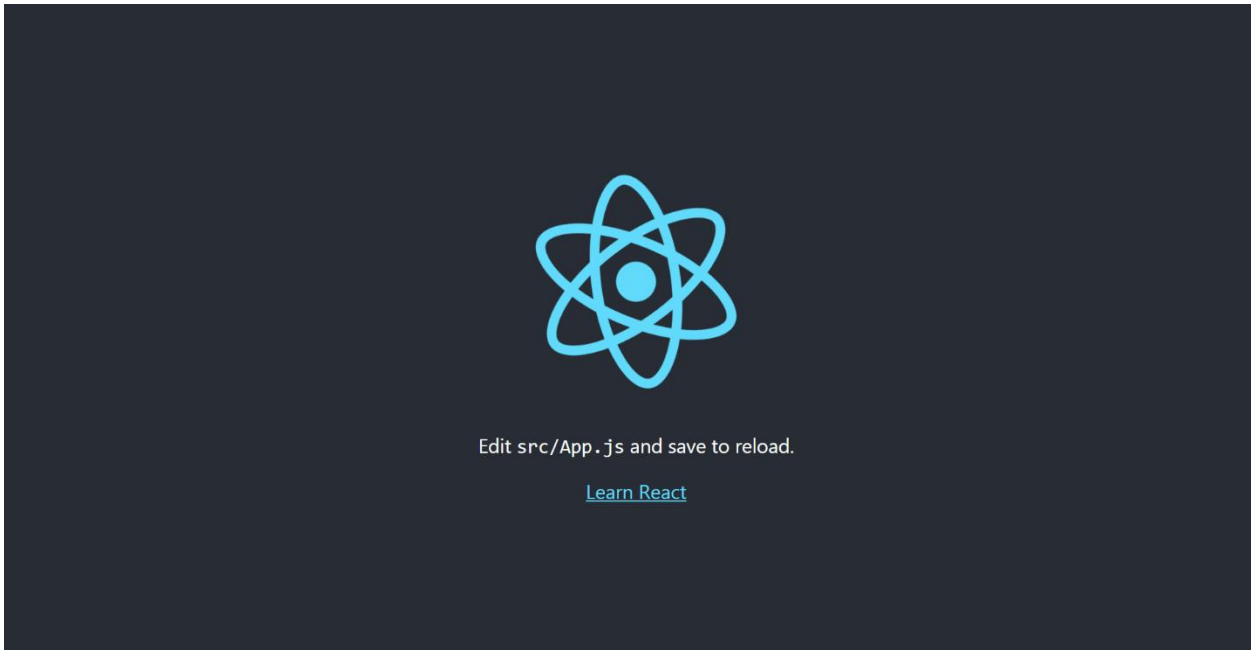


Рис. 3.3. Вікно браузера з проєктом

Повертаємось до терміналу та послідовно вводимо команди npm install electron electron-builder wait-on concurrently та npm install electron-is-dev. Ці команди додають до проєкта модуль Electron. Тепер створюємо в папці public файл main.js з наступним кодом:

```
const electron = require('electron');
const app = electron.app;
const BrowserWindow = electron.BrowserWindow;

const path = require('path');
const url = require('url');
const isDev = require('electron-is-dev');

let mainWindow;

function createWindow() {
  mainWindow = new BrowserWindow({width: 900, height: 680});
  mainWindow.loadURL(isDev ? 'http://localhost:3000' : `file://$
{path.join(__dirname, '../build/index.html')}`);
  mainWindow.on('closed', () => mainWindow = null);
}

app.on('ready', createWindow);

app.on('window-all-closed', () => {
  if (process.platform !== 'darwin') {
    app.quit();
  }
});

app.on('activate', () => {
  if (mainWindow === null) {
    createWindow();
  }
});
```

Таким чином тепер існує файл, який буде запускати програму замість початкового index.js(Рис. 3.4). Для того, щоб це відбувалось, треба додати до файлу package.json рядок "main": "public/main.js". Залишилося додати скрипт, який буде відкривати в вікні локальний порт з шаблоном проєкту React. Для цього в package.json до скритів треба додати

"electron-dev": "concurrently \"BROWSER=none npm run start\" \"wait-on http://localhost:3000 && electron .\""

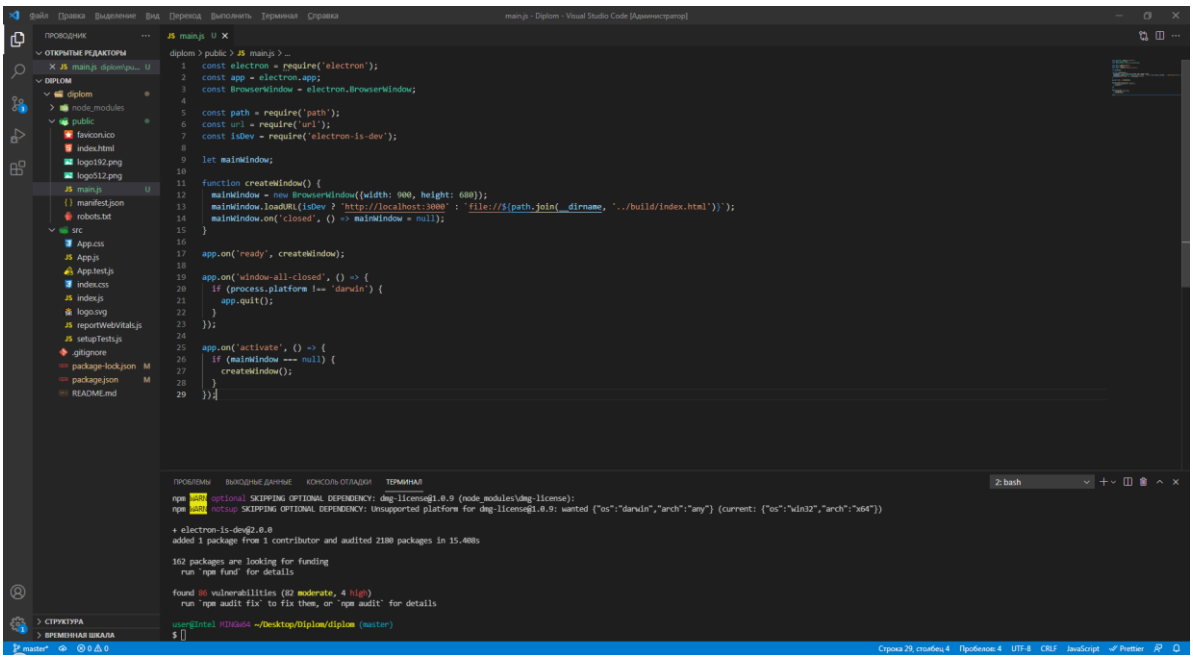


Рис. 3.4. Створений файл main.js

Остаточний вигляд файлу package.json:

```
{
  "name": "diplom",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.13.0",
    "@testing-library/react": "^11.2.7",
    "@testing-library/user-event": "^12.8.3",
    "concurrently": "^6.2.0",
    "electron": "^13.1.2",
    "electron-builder": "^22.11.7",
    "electron-is-dev": "^2.0.0",
    "react": "^17.0.2",
    "react-dom": "^17.0.2",
    "react-scripts": "4.0.3",
    "wait-on": "^5.3.0",
    "web-vitals": "^1.1.2"
  },
  "author": "Andrii Hrytsyna",
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject",
    "electron-
dev": "concurrently \"BROWSER=none npm run start\" \"wait-
on http://localhost:3000 && electron .\""
```

```
  },
  "main": "public/main.js",
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

Тепер можна запустити програму командою npm run electron-dev та побачити теж саме вікно React, але у вікні(Рис. 3.5).

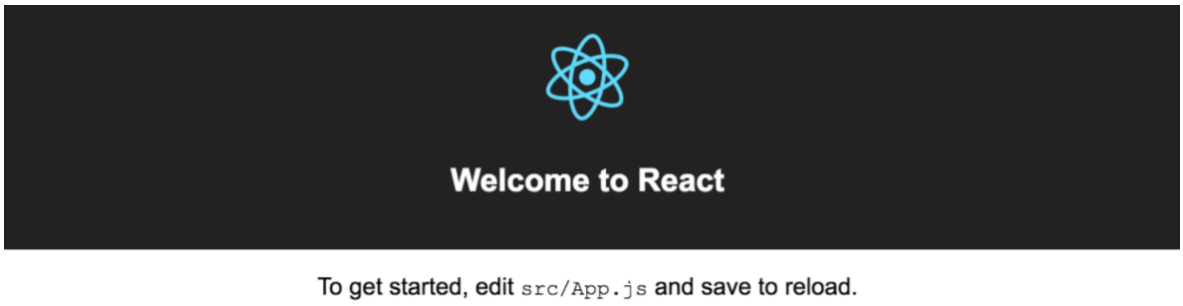


Рис. 3.5. Сторінка React у вікні

3.2 Створення загального функціоналу

Перш за все треба створити меню, завдяки якому користувач зможе орієнтуватись у додатку. Для цього робоча область поділяється на дві частини, завдяки Sass можна точно виміряти скільки процентів сторінки відходить тому чи іншому елементу. Потім створюються умовні зони посилань на теорію та на практичну реалізацію. Схематичний вигляд представлений на Рис. 3.6.

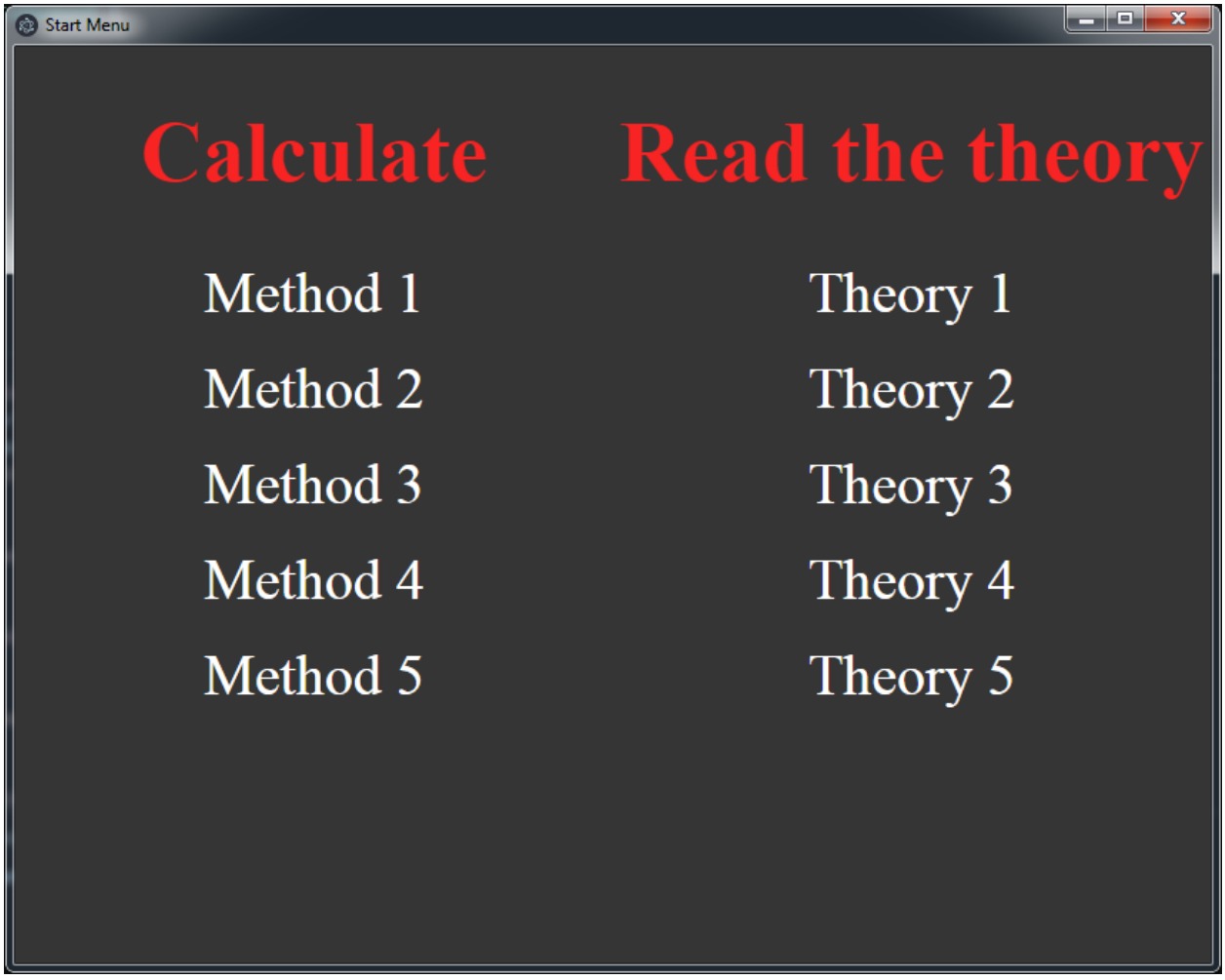


Рис. 3.6. Схематичне меню додатку

Для того, аби меню працювало, потрібно налаштувати Switch- клас, в якому завдяки Route path можна вказувати яку саме компоненту треба відобразити при переході на піддомен. Налаштовуємо кожне посилання на відповідну компоненту. Склад кожної компоненти з калькулятором однакова: є поля для вводу, кнопка для обробки та результат(Рис. 3.7).

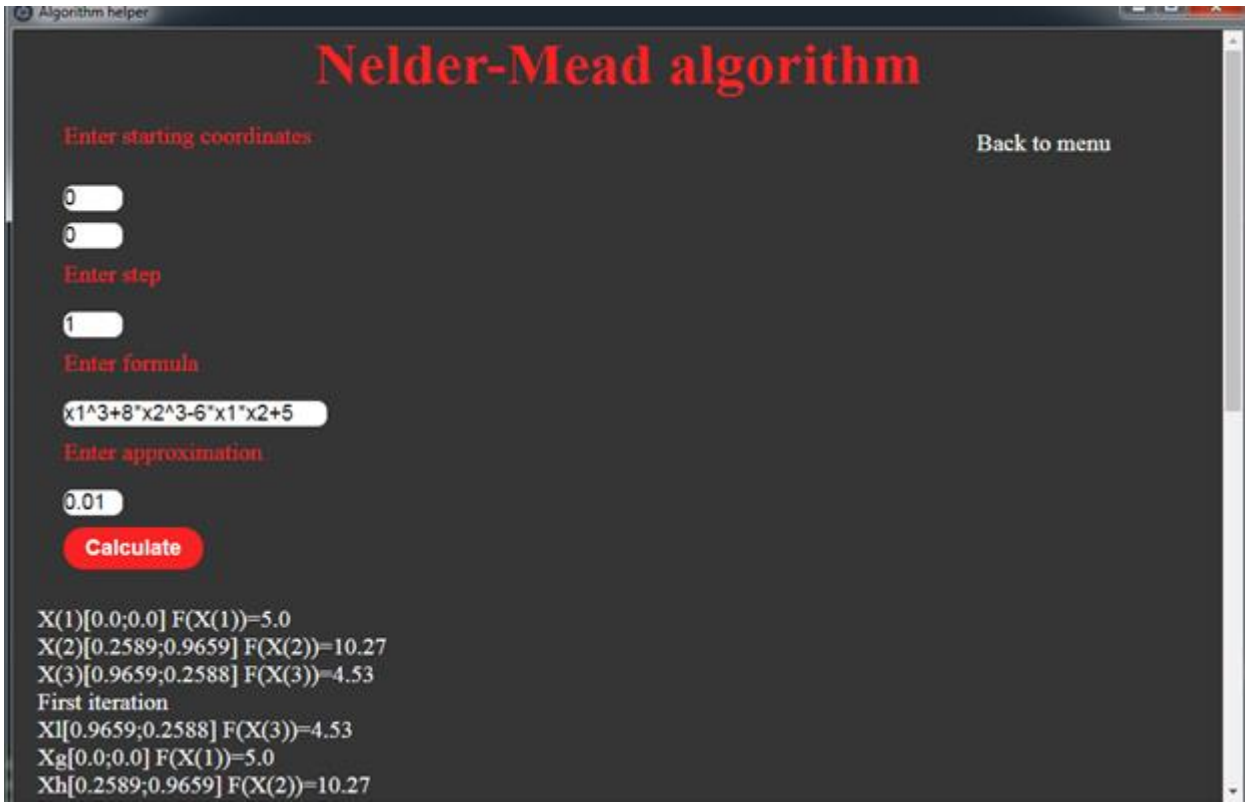


Рис. 3.7. Компонента з алгоритмом Нелдера-Міда

Для інтернаціоналізації використовується бібліотека i18n. Після завантаження її модуля, необхідно додати у src папку файл i18n,js(Рис. 3.8).

```
JS i18n.js > ...
import i18n from 'i18next';
import { initReactI18next } from 'react-i18next';

import Backend from 'i18next-http-backend';
import LanguageDetector from 'i18next-browser-languagedetector';
// don't want to use this?
// have a look at the Quick start guide
// for passing in lng and translations on init

i18n
  // load translation using http -> see /public/locales (i.e. https://github.com/i18next/react-i18next/tree/master/example/react/public/locales)
  // learn more: https://github.com/i18next/i18next-http-backend
  .use(Backend)
  // detect user language
  // learn more: https://github.com/i18next/i18next-browser-languageDetector
  .use(LanguageDetector)
  // pass the i18n instance to react-i18next.
  .use(initReactI18next)
  // init i18next
  // for all options read: https://www.i18next.com/overview/configuration-options
  .init({
    fallbackLng: 'en',
    debug: true,

    interpolation: {
      escapeValue: false, // not needed for react as it escapes by default
    }
  });


export default i18n;
```

Рис. 3.8. Файл i18n.js

Після цього треба імпортувати цей скрипт в index.js та створити папки з перекладами public/translation/en та public/translation/ua. За необхідністю можна додати й інші мови, так як принцип роботи перекладу в формуванні json файлів з зразками, які потребують перекладу та його варіанту на відповідній мові. Для зручного перемикання між мовами створено віджет з прапорами країн. Меню англійською буде мати вигляд, зазначений на Рис. 3.9., українською – на Рис. 3.10.

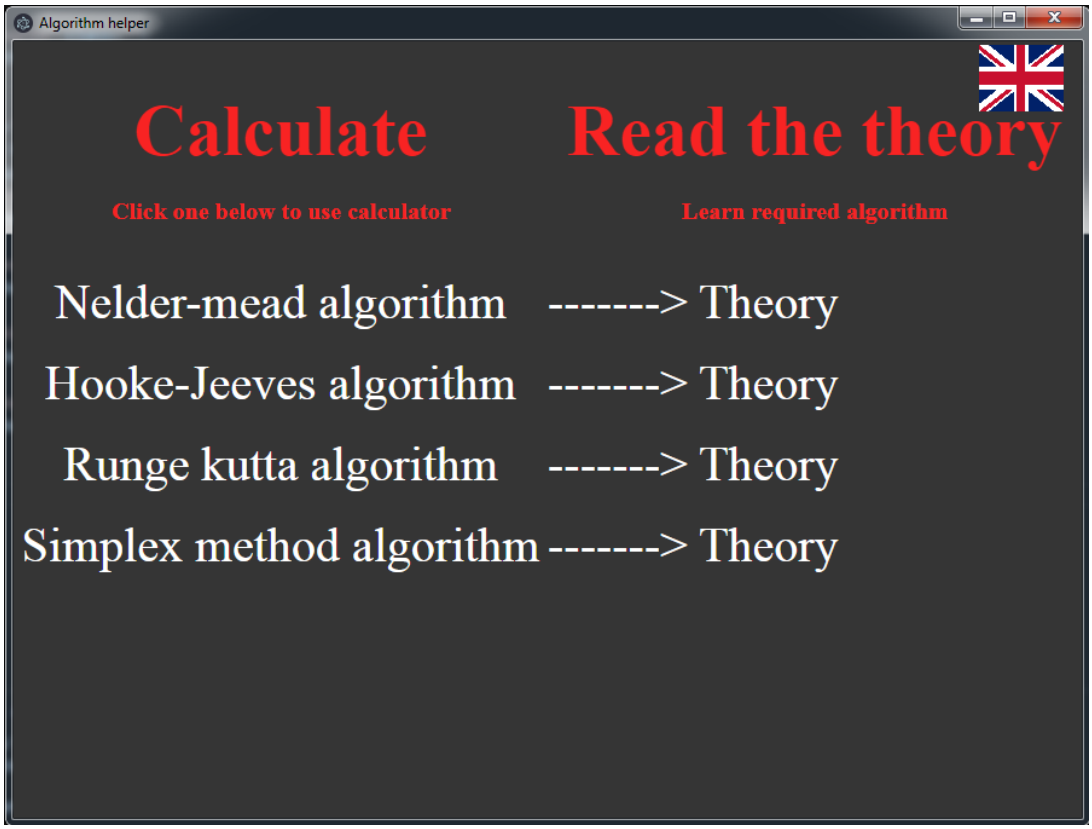


Рис. 3.9. Меню додатку англійською мовою

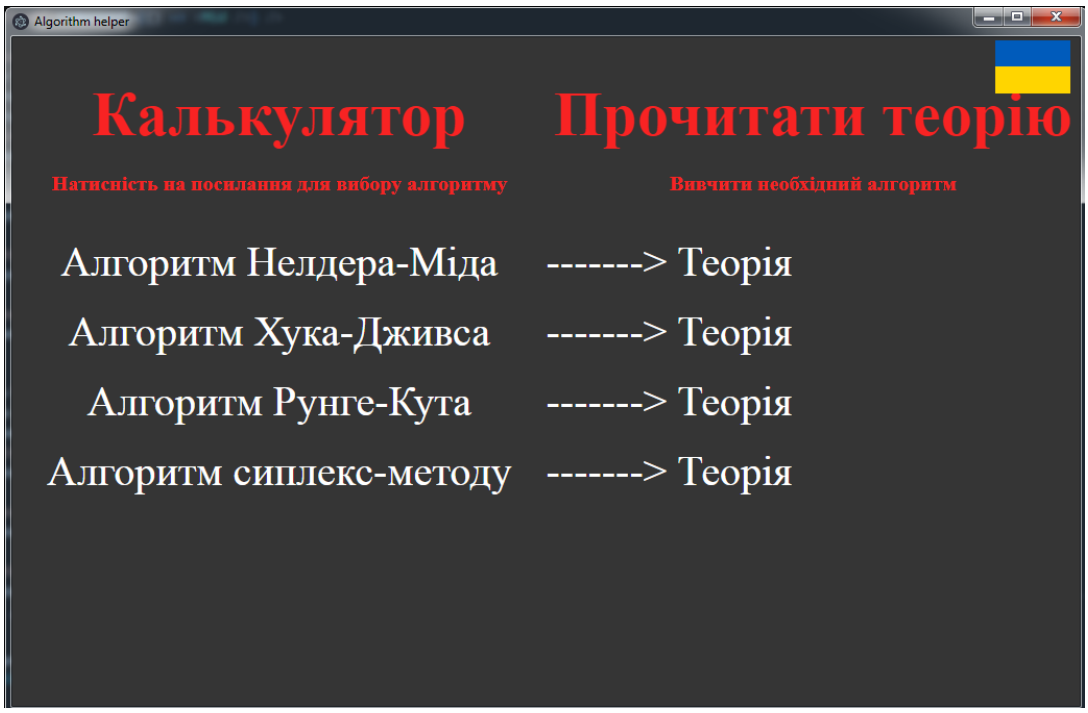


Рис. 3.10. Меню додатку українською мовою

ВИСНОВОК

У кваліфікаційній бакалаврській роботі проаналізована проблематика створення додатку для реалізації алгоритмів чисельних методів за допомогою веб-технологій. Було проведено дослідження ринку, на основі якого був вибраний тип майбутнього додатку. Досліджені основні концепції, структура та архітектура створюваного додатку, завдяки чому були сформовані основні вимоги до розроблюваного продукту. Для реалізації було обрано фреймворки React разом з Electron. Розроблена логіка роботи додатку, для кращого розуміння якої розроблені Use-case діаграма, діаграма активностей та DFD діаграми троьх рівнів. Розробка почалася з інтеграції шаблону проєкту React у формат, який надає Electron. Створення подальшого функціоналу призначене для реалізації алгоритмів чисельних методів та відповідності функціональних та нефункціональних вимог. Ключові етапи створення додатку наведені в останньому пункті роботи. В результаті роботи виконані всі вимоги які ставилися на етапі дослідження та розробки логіки, додаток протестовано, лістинг частин коду знаходиться у ДОДАТОК. Оскільки додаток направлений на навчання, то проєкт можна розширити, інтегрувавши у системи змішаного навчання Сумського Державного Університету.

СПИСОК ЛІТЕРАТУРИ

1. Найпопулярніші професії ринку праці України [Електронний ресурс] – Режим доступу: https://mik.dcz.gov.ua/publikaciya/naypopulyarnishi-profesiyi-rynku-praci-ukrayiny.
2. Всеукраїнська науково-практична Інтернет-конференція «Автоматизація та комп'ютерно-інтегровані технології у виробництві та освіті: стан, досягнення, перспективи розвитку», 2015 р., м. Черкаси – 276с.
3. What is a Web Application? [Електронний ресурс] – Режим доступу: https://blog.stackpath.com/web-application/
4. What Is a Web Application? How It Works, Benefits and Examples [Електронний ресурс] – Режим доступу: https://www.indeed.com/career-advice/career-development/what-is-web-application.
5. Тренды на рынке труда 2021 в ИТ [Електронний ресурс] – Режим доступу: https://hr-elearning.ru/trendy-na-rynke-truda-2021-v-it/.
6. Как COVID-19 изменил рынок труда: зарплаты, вакансии, фрилансеры и прогнозы на 2021 год [Електронний ресурс] – Режим доступу: https://www.epravda.com.ua/rus/publications/2021/01/4/669724/.
7. Що таке корпоративний сайт? [Електронний ресурс] – Режим доступу: https://www.centum-d.com/uk/shho-take-korporativnij-sajt/.
8. PWA — это просто [Електронний ресурс] – Режим доступу: https://habr.com/ru/post/418923/.
9. Что такое edtech, почему учебные онлайн-курсы для детей должны быть доступны каждому и зачем учиться программировать с начальных классов? [Електронний ресурс] – Режим доступу:

https://www.everest.ua/ru/chto-takoe-edtech-pochemu-uchebnye-onlajn-kursy-dlya-detej-dolzhny-byt-dostupny-kazhdomu-y-zachem-uchytsya-programmyrovat-s-nachalnyh-klassov/.

10. Проектирование и разработка Web-приложений / А. Ф. Тузовский – Москва, 2019 – 218с.
11. Web Protocols and Practice: Http/1.1, Networking Protocols, Caching, and Traffic Measurement, 1st Edition / B. Krishnamurthy , J. Rexford - Addison-Wesley,2001 – 672с.
12. Advantages and Disadvantages of Microservices Architecture [Електронний ресурс] – Режим доступу: https://cloudacademy.com/blog/microservices-architecture-challenge-advantage-drawback/.
13. State of Frontend 2020 [Електронний ресурс] – Режим доступу: https://tsh.io/state-of-frontend/#frameworks/.
14. React: Up & Running, 2nd Edition / S. Stefanov - O'Reilly, 2021 – 222с.
15. Components and Props [Електронний ресурс] – Режим доступу: https://reactjs.org/docs/components-and-props.html/.
16. React in Action, 1st Edition / M. T. Thomas – Manning Shelter Island, 2019 – 366с.
17. Electron vs PWA: The Pros And Cons Of Both Approaches [Електронний ресурс] – Режим доступу: https://javascript.plainenglish.io/electron-vs-pwa-the-pros-and-cons-of-both-approaches-b4ce172f0022.
18. Electron in Action, 1st Edition / S. Kinney - Manning Publications, 2018 – 376с.
19. Основы Sass [Електронний ресурс] – Режим доступу: https://sass-scss.ru/guide/.
20. What Is i18n? [Електронний ресурс] – Режим доступу: https://lingoport.com/what-is-i18n/.

ДОДАТОК

App.jsx

```
import './App.sass'
import Runge from '../components/Runge/Runge
import Simple from '../components/Simple/Simple
import Jives from '../components/Jives/Jives
import Nelder from '../components/Nelder/Nelder
class App extends React.Component {
    state = {
    }


    render ()
    {
      return (
        <Suspense fallback="loading">
            <Router>
                <Switch>
                    <Route path='/nelder component={()=><Nelder>
} />
                    <Route path='/jives component={()=><Jives/>
}/>
                    <Route path='/simple component={()=><
Simple />} />
                    <Route path='/runge component={()=><Runge>}
/>
                </Switch>

            </Router>
            <ToastContainer/>
            </Suspense>

        );
    }
}

export default App
```

Nelder.jsx

```
class App extends React.Component {
  state = {};

  render() {
    return (
      <div className="container">
        <h1>Nelder-Mead algorithm</h1>
```

```
        <div className="lower-container">
          <p>Enter starting coordinates</p>
          <div classname="dots">
            <input className="dot"></input>
            <input className="dot"></input>
          </div>
          <p>Enter step</p>
          <input className="dot"></input>
          <p>Enter formula</p>
          <input></input>
          <p>Enter approximation</p>
          <input className="dot"></input>
          <button className="btn">Calculate</button>
        </div>
        <div className = "result">
     var solution = fmin.nelderMead(loss, [-3.5, 3.5]);
return Math.sin(y) * x  + Math.sin(x) * y  +  x * x +  y *y;
     }
        </div>


<Router>
        <Link to='/>Back to menu</Link>
        </Router>
      </div>
    );
  }
}

export default Nelder;
```

Nelder.sass

```
*
  text-decoration: none
  box-sizing: border-box
  border: none
  padding: 0
  margin: 0

body
  min-height: 100vh
  background-color: #353535
  color: #fff

.container
  &-shap
    display: flex
```

```
    flex-direction: row
    margin-top: 40px
    h1
      text-align: center
      margin-bottom: 20px
      font-size: 60px
      letter-spacing: 0px
      color: #f82323
    h2
      text-align: center
      margin-bottom: 20px
      font-size: 20px
      letter-spacing: 0px
      color: #f82323
    a
      padding-top: 20px
      font-size: 40px
      color: #fff
      &:hover
        color: #f82323

  &-calculate
    width: 50%
    display: flex
    flex-direction: column
    text-align: center
  &-theory
    width: 50%
    display: flex
    flex-direction: column
    text-align: left

h1
  text-align: center
  margin-bottom: 20px
  font-size: 50px
  letter-spacing: 0px
  color: #f82323
input

  display: flex
  flex-direction: column
  margin-top: 10px
  margin-bottom: 10px
  font-size: 18px
  background: #fff
  border-radius: 8px
  color: #000
input.dot
  width: 5%
```

```
.dots
  display: flex
  flex-direction: column
.btn
  background: #f82323
  padding: 7px 18px
  font-size: 18px
  border-radius: 18px
  color: #fff
  font-weight: bold
  cursor: pointer

  transition: 0.3s
.lower-container
  padding-left: 50px
  p
  margin-bottom: 30px
  font-size: 30px
  color: #f82323
.result
  p
  margin-left: 20px
  margin-bottom: 20px
  font-size: 20px
  color: #fff

a
  position: absolute
  top: 10%
  right: 10%
  padding-top: 20px
  font-size: 20px
  color: #fff
  &:hover
    color: #f82323
```

Runge.jsx

```
class App extends React.Component {
  state = {};

  render() {
    return (
      <div className="container">
        <h1>Runge-Kutta algorithm</h1>
        <div className="lower-container">
          <p>Enter starting data</p>
          <div classname="dots">
            <input className="dot"></input>
            <input className="dot"></input>
```

```
          </div>
          <p>Enter approximation</p>
          <input className="dot"></input>
          <button className="btn">Calculate</button>
        </div>
        <div className = "result">
const dSIR = (t, y) => [-
T * y[0] * y[1], (T * y[0] - R) * y[1], R * y[1]];

// Solve the system and log the result (reduced to the infection
 count).
console.log(rungeKutta(dSIR, [1, .1, 0], [0, 14], .2).map(x => x
[1]));
<Router>
        <Link to='/>Back to menu</Link>
        </Router>
      </div>
    );
  }
}

export default Runge;
```

i18n.js

```
// append app to dom
ReactDOM.render(
  <App />,
  document.getElementById("root")
);
```

Translation/en/json.

```
{
    "menu": {
        "Calculate": "Calculate",
        "Click below": "Click one below to use calculator",
        "Read the theory": "Read the theory",
        "Learn": "Learn required algorithm",
        "Theory": "Theory",
        "Nelder": "Nelder-mead algorithm",
        "Jeeves": "Hooke-Jeeves algorithm",
        "Runge": "Rungt kutta algorithm",
        "Simplex": "Simplex method algorithm",
}
}
```

Translation/ua/json.

```
{
    "menu": {
```

```
        "Calculate": "Калькулятор",
        "Click below": "Натисніть на посилання для вибору
алгоритму",
        "Read the theory": "Прочитати теорію",
        "Learn": "Вивчити необхідний алгоритм",
        "Theory": "Теорія",
        "Nelder": "Алгоритм Нелдера-Міда",
        "Jeeves": "Алгоритм Хука-Дживса",
        "Runge": "Алгоритм Рунге-Кута",
        "Simplex": "Алгоритм симплекс методу",
}
}
```