МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Реінжиніринг інформаційної системи автоматизованої перевірки лабораторних робіт з вибірки даних»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ.м-01 Давиденко Данило Андрійович

Кваліфікаційну роботу захищена на засіданні ЕК з оцінкою _____ «__» грудня 2021 р.

Науковий керівник _____ к.т.н., доц., Марченко А. В.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Голова комісії _____ Шифрін Д. М.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2021

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В. В. Шендрик
«__» _____ 2021 р.

ЗАВДАННЯ
на кваліфікаційну роботу магістра студентові

Давиденко Данило Андрійович

1 Тема роботи *Реінжиніринг інформаційної системи автоматизованої перевірки лабораторних робіт з вибірки даних*

затверджена наказом по університету від « 29 » жовтня 2021 р. № 0787-VI

2 Термін здачі студентом закінченого проекту « 22 » грудня 2021 р.

3 Вхідні дані до роботи технічне завдання на реінжиніринг web-орієнтованої системи, методичні вказівки до дисциплін «Організація баз даних і баз знань».

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, постановка задачі та методи дослідження, проектування інформаційної системи автоматизованої перевірки лабораторних робіт з вибірки даних, розробка інформаційної системи автоматизованої перевірки лабораторних робіт з вибірки даних.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність проблеми, аналіз аналогічних рішень, порівняльна таблиця, постановка задачі, контекстна діаграма, діаграма декомпозиції першого рівня, діаграма декомпозиції другого рівня, ER-діаграма бази даних системи, діаграма моделі бази знань, ER-діаграма навчальної бази даних, діаграма варіантів використання, архітектура програмного забезпечення, фрагменти лістингу коду програмної реалізації, приклад використання додатку.

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання 25.10.2021.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Планування робіт проекту	26.10.2021 – 31.10.2021	
2.	Огляд останніх досліджень і публікацій	01.11.2021 – 03.11.2021	
3.	Аналіз аналогічних рішень	05.11.2021 – 06.11.2021	
4.	Постановка задачі проекту	07.11.2021 – 08.11.2021	
5.	Функціональне моделювання бізнес-процесів	09.11.2021 – 11.11.2021	
6.	Проектування концепції системи	12.11.2021 – 13.11.2021	
7.	Проектування моделей бази даних та бази знань	14.11.2021 – 16.11.2021	
8.	Розробка web-застосунку	17.11.2021 – 05.12.2021	
9.	Тестування web-застосунку	06.12.2021 – 08.12.2021	
10.	Оформлення документації	09.12.2021 – 12.12.2021	
11.	Здача пояснювальної записки	13.12.2021	

Магістрант _____ Давиденко Д.А.
(підпис)

Керівник роботи _____ к.т.н., доц. Марченко А.В.
(підпис)

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Реінжиніринг інформаційної системи автоматизованої перевірки лабораторних робіт з вибірки даних».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 18 найменувань, додатків. Загальний обсяг роботи − 126 сторінки, у тому числі 60 сторінок основного тексту, 3 сторінки списку використаних джерел, 63 сторінка додатків.

Кваліфікаційну роботу магістра присвячено реінжинірингу web-орієнтованої інформаційної системи автоматизованої перевірки лабораторних робіт з вибірки даних.

В роботі проведено аналіз актуальності реінжинірингу наявної системи, огляд останніх досліджень і публікацій, аналіз аналогічних рішень, проектування системи.

У роботі виконано розробку бази даних системи та бази знань для перевірки та оцінювання відповідей студентів, а також web-застосунку, що включає ряд функціональних модулів:

- Модуль авторизації/реєстрації;
- Модуль навігації між сторінками системи;
- Модуль роботи з інформацією про лабораторні роботи та їх завдання;
- Модуль роботи з інформацією про користувачів системи;
- Модуль роботи з відповідями студентів та, встановленими викладачем, правильним відповідями на завдання;
- Модуль виконання та перевірки завдань з лабораторних робіт.

Результатом проведеної роботи є модернізована web-орієнтована інформаційна система автоматизованої перевірки лабораторних робіт з вибірки даних.

Ключові слова: лабораторна робота, перевірка завдань, база даних, SQL, вибірка даних, MVC, web-застосунок.

ЗМІСТ

ВСТУП

Підходи до розробки програмного забезпечення (ПЗ) для інформаційних систем (ІС) безперервно розвивається: з’являються нові, покращуються наявні. Тому практично неможливо створити ІС, яка матиме актуальне ПЗ впродовж тривалого проміжку часу. Проте, кожен власник ІС хоче, щоб користувачі постійно отримували найкращій досвід при використанні його системи та продовжували нею користуватися. Для цього потрібно періодично вдосконалювати її згідно з актуальними стандартами сфери інформаційних технологій, а також надавати користувачам зручний новітній функціонал, що вирізнятиме систему з-поміж аналогічних рішень.

Головна мета проекту полягатиме у модернізації поточної архітектури ПЗ системи, та вдосконаленні програмного модуля автоматизованої перевірки завдань з лабораторних робіт.

Однією з основних задач, що постане при реінжинірингу ІС є проведення зворотної розробки для визначення поточних принципів функціонування системи. Не менш важливим є вибір найвідповіднішого архітектурного шаблону ПЗ, та реформація наявних програмних модулів згідно схеми його структурної організації. Покращення автоматизованої роботи модуля перевірки завдань з лабораторних робіт передбачатиме впровадження відповідної підсистеми, яка буде визначати правильність, наданих студентами, відповідей. Для безпомилкової роботи цієї системи необхідно обрати найбільш відповідній алгоритм синтаксичного аналізу SQL запитів, та створити базу знань, що дозволить проводити оцінку відповідності рішень студентів до правильних відповідей, визначених викладачем. Ще однією задачею є поліпшення наявного користувальницького інтерфейсу задля підвищення ефективності роботи з ІС, а також збереження прийнятної швидкодії системи шляхом оптимізації роботи програмних модулів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Відповідно до мети та задач проекту було виділено три основних теми, що потребують детального вивчення шляхом розгляду актуальних літературних джерел: реінжиніринг ПЗ, синтаксичний аналіз SQL запитів та проектування бази знань для підсистеми перевірки завдань. Далі, наведено огляд досліджень і публікацій за кожною з визначених тем.

У статті «Awareness Driven Software Reengineering» [1] розглядаються передумови, способи та основні етапи проведення реінжинірингу ПЗ. У своїй роботі автор приділяє увагу правильності аналізу застарілого вихідного коду та потребі його рефакторингу. Також, зазначається необхідність використання принципу розподілу відповідальності для побудови адаптивних систем. Ще однією публікацією є монографія «Реінжиніринг програмного забезпечення інформаційних систем» [2], в якій описано загальні принципи реінжинірингу, а крім того, наводиться приклад аналіз програмного коду модулів інформаційної системи за допомогою лінгвістичних методів. У цій роботі, також, розповідається про парадигму породжувальних граматик [3] для моделювання лінгвістичного забезпечення та технологію мультилінгвістичного перекодування ПЗ.

Стаття «Design and implementation of general SQL parser» [4] містить відомості про способи лексичного та синтаксичного аналізу SQL запитів до бази даних, а ще у ній розповідається про оптимізацію синтаксичного дерева, побудованого під час розкладання запиту на лексеми, та виявлення помилок у структурі запитів. У статті «Синтаксичний аналізатор мови SQL» [5] автор наводить класифікацію способів синтаксичного аналізу за типом мети, синтаксичною структурою та формальною теорією опису природної мови, і описує особливості кожного з підходів.

«Automated Expert System Knowledge Base Development Method for Information Security Risk Analysis» [6] це стаття, у якій йдеться про метод створення (наповнення) бази знань за допомогою експертної системи, яка проводить перетворення існуючої онтології у правила бази знань. У ній детально розглядається алгоритм процесу трансформації логіки правил написаних на мові правил онтології (OWL) у формат обміну правилами (RIF) та його трансформацію у правила бази знань, а також переваги використання цього підходу. У роботі «Розробка навчальної експертної системи для компаративного аналізу художніх творів» [7] автор розповідає про створення програмного засобу, що використовує експертну систему для порівняльного аналізу художнього твору, проводить огляд його функціональних особливостей та структурних елементів.

В результаті огляду зазначених джерел було визначено найбільш вдалий підхід до проведення реінжинірингу ПЗ системи, що полягає у поетапній зворотній розробці та вибору архітектурного шаблону, що дозволяє реалізувати принцип розподілу відповідальності; сформовано представлення про необхідний алгоритм для аналізу SQL запитів, що має передбачати розкладання запиту на ряд компонентів для більш детального аналізу його семантики; окреслено необхідний формат бази знань, що представлятиме собою набір записів, які представлятимуть собою набір окремих компонентів для кожної правильної відповіді.

1.2 Аналіз продуктів-аналогів

Серед програмних продуктів, що виконують задачі, схожі до поставлених у даному проекті, можна виділити ряд web-орієнтованих рішень для вивчення мови SQL. Всі вони передбачають виконання практичних завдань шляхом написання запитів до тренувальної бази даних.

Інтерактивний онлайн тренажер мови SQL – SQLCourse [8]. Вигляд сторінки тренування навичок створення вибірки даних для цього онлайн тренажера наведено на рисунку 1.1.

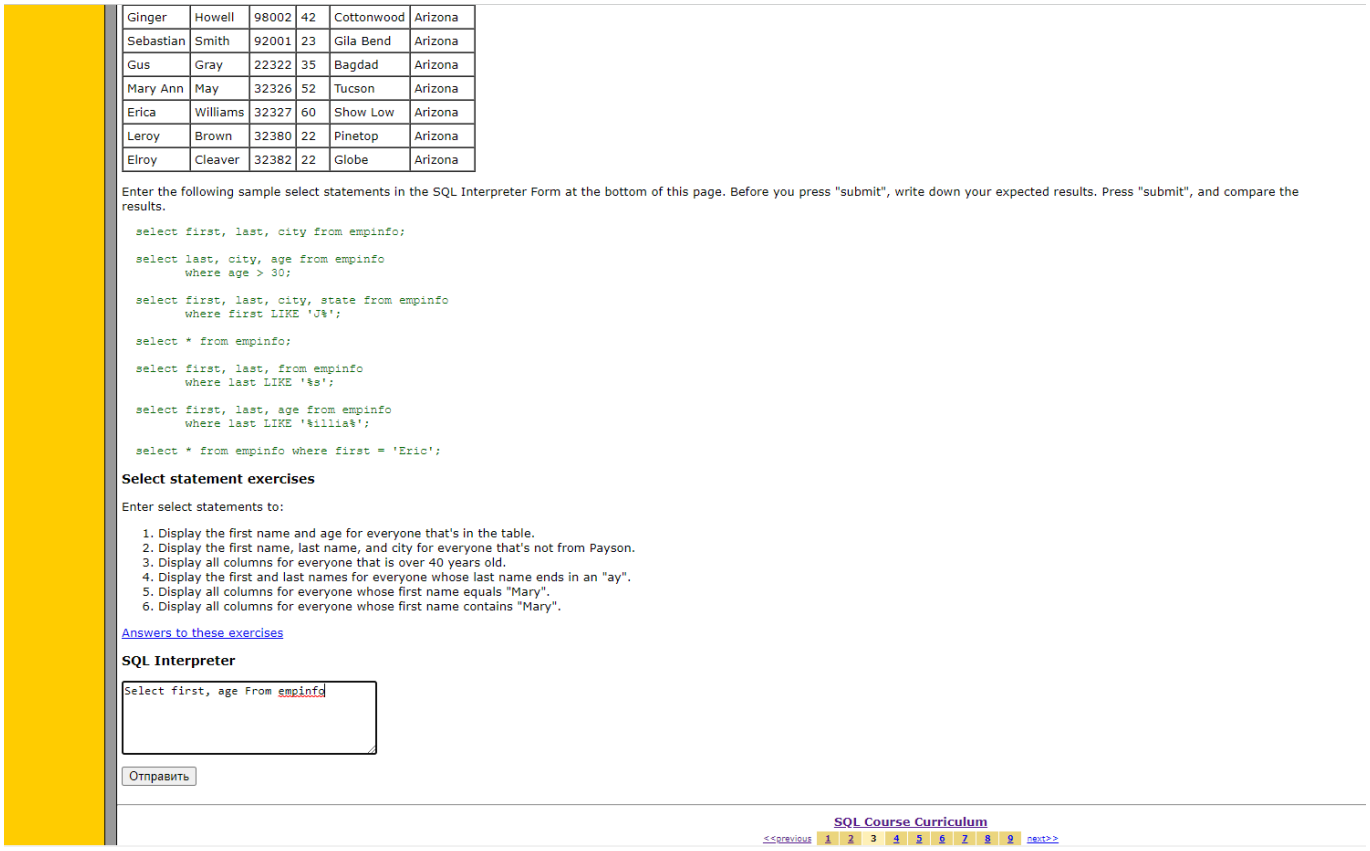


Рисунок 1.1 – Сторінка для тренування навичок створення вибірки даних SQLCourse

За своєю структурою SQLCourse нагадує підручник для вивчення мови SQL та ряду систем управління базами даних. Користувач може обрати певну тему, потім переглянути теоретичний матеріал та приклади побудови запитів за цією темою, а в кінці створити запит до тестової бази даних, і отримати певні практичні навички. Інтерфейс даного онлайн тренажера виглядає застаріло, проте він доволі простий, тому знаходження необхідних опцій не викликає проблем. Функціональний модуль, який використовується для перевірки, введених користувачем, запитів та виводу результатів на екран має ряд проблем і не завжди відображає отриманий результат у відповідному форматі.

Інтерактивний онлайн тренажер мови SQL – SQLZoo [9]. Вигляд сторінки тренування навичок створення вибірки даних для цього онлайн тренажера наведено на рисунку 1.2.

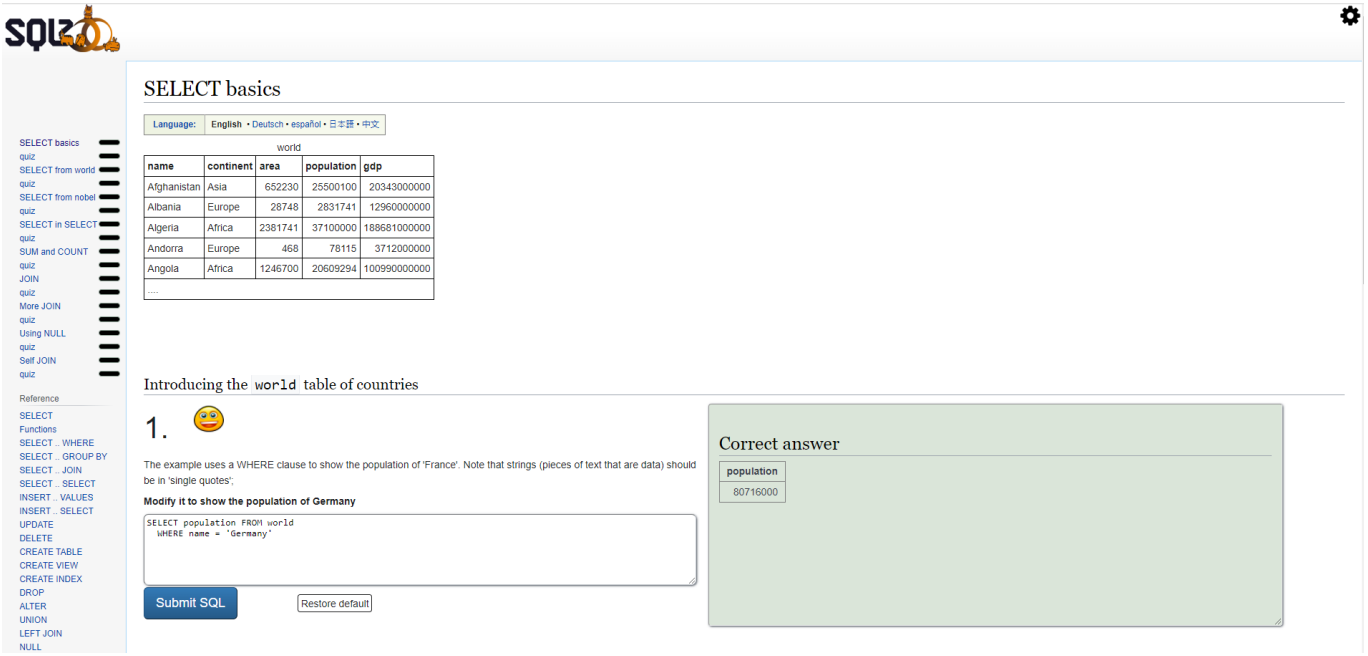


Рисунок 1.2 – Сторінка для тренування навичок створення вибірки даних SQLZoo

SQLZoo представляє собою онлайн платформу для відпрацювання практичних навичок написання запитів до тестової бази даних. Процедура використання цього тренажера передбачає перегляд таблиці даних, що використовується у запитах, написання запиту за визначеним завданням та отримання оцінки правильності результату. Також, після кожної секції із завданнями з написання запитів, користувачу пропонується пройти вікторину за темою попередньої секції. Інтерфейс платформи простий, дружній та добре узгоджений. Модуль для написання запитів та відображення результату працює швидко та надає змістовну інформацію про результат виконання запиту.

Платформа для оцінювання навичок спеціалістів HakerRank [10]. Вигляд сторінки тренування навичок створення вибірки даних для цієї платформи наведено на рисунку 1.3.

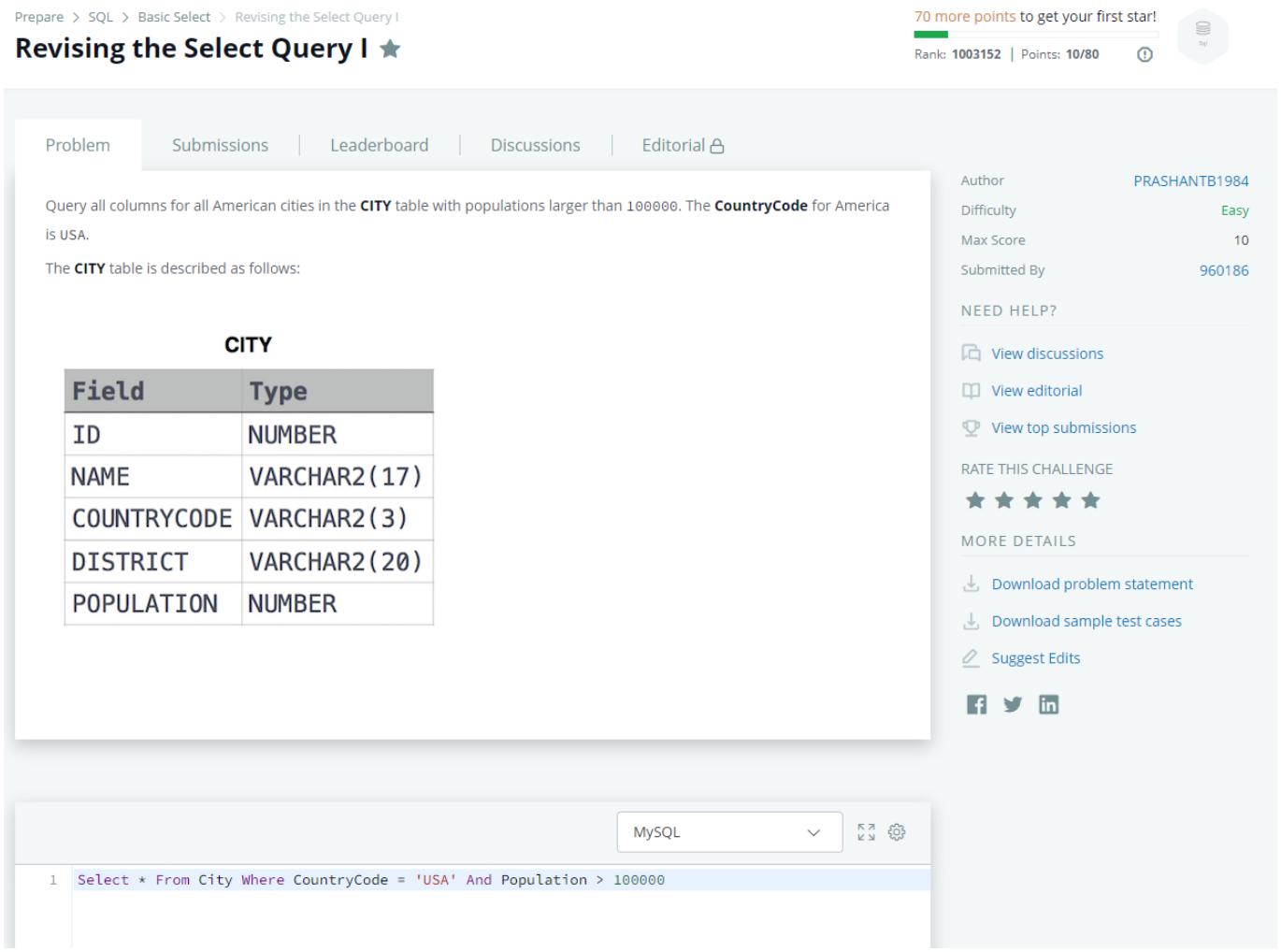


Рисунок 1.3 – Сторінка для тренування навичок створення вибірки даних HakerRank

HakerRank – це високотехнологічна платформа, яку багато компаній використовують для найму розробників, а розробники – для демонстрації своїх технічних навичок. Дана платформа надає дуже широкий спектр завдань для різноманітних технологій. При виконанні завдання за тематикою мови SQL користувачу надається завдання та структура таблиці, яку він має використовувати у своєму запиті. Також, можна переглянути результати своїх попередніх спроб для даного завдання, дошку лідерів (хто найкраще виконав завдання) та обговорення до цього завдання. Інтерфейс дуже зручний та дозволяє ефективно працювати з платформою. При використання функціонального модуля для написання запитів та отримання результатів можна обрати систему управління базами даних, встановити налаштування текстового редактора, завантажити код з файлу та виконувати запити.

При цьому наявна опція попереднього виконання коду, тобто, користувач має змогу спочатку переглянути очікуваний результат, який він отримає після виконання запиту, а вже потім підтвердити свою відповідь. Відображення результату запиту відбувається у зручному форматі, проте не завжди можна чітко зрозуміти проблему у разі виникнення помилки при виконанні.

Інтерактивний онлайн тренажер мови SQL – SQLBolt [11]. Вигляд сторінки тренування навичок створення вибірки даних для цього онлайн тренажера наведено на рисунку 1.4.

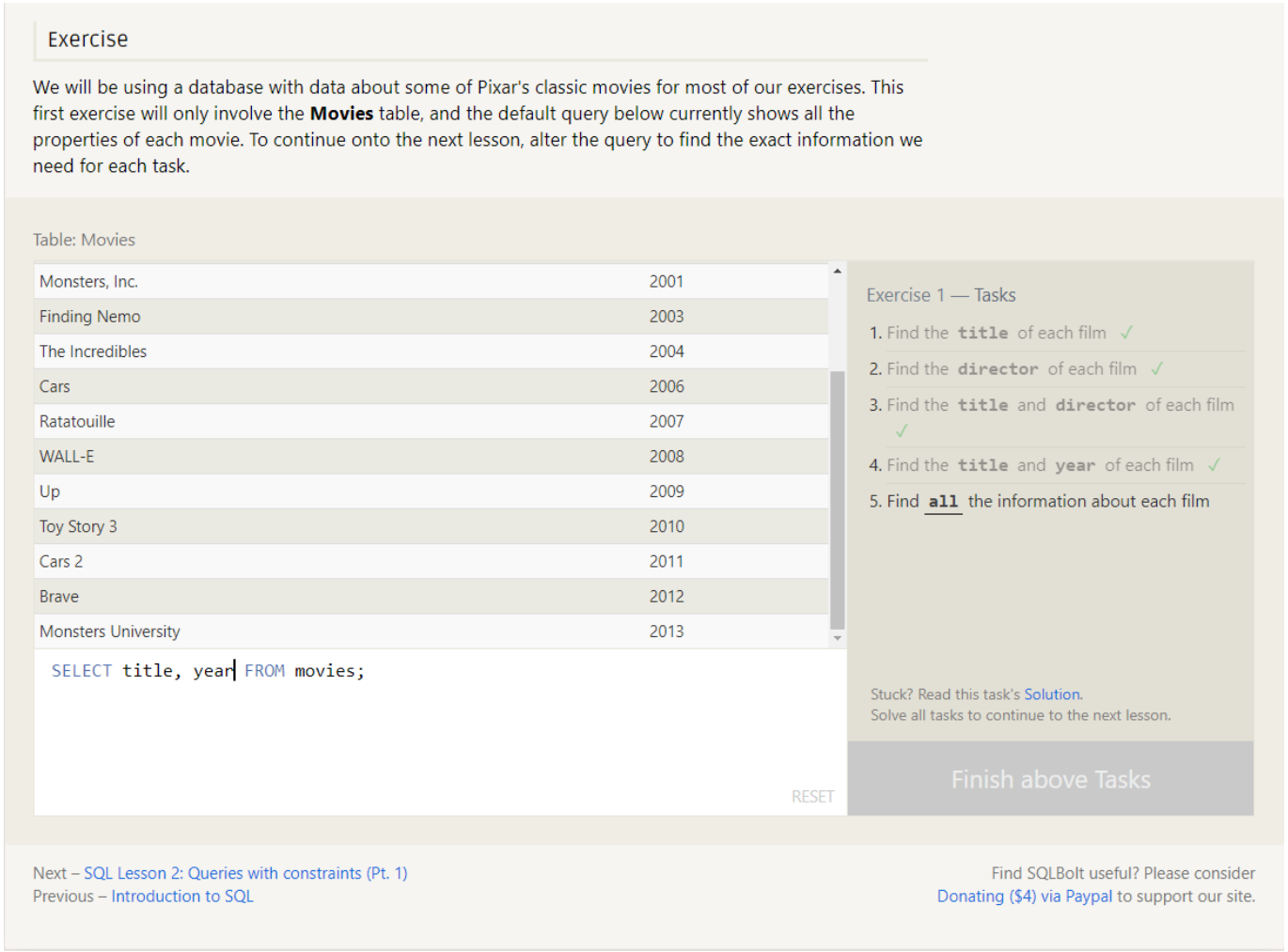


Рисунок 1.4 – Сторінка для тренування навичок створення вибірки даних SQLBolt

SQLBolt – це ще одне рішення, яке за своєю структурою нагадує підручник, що містить набір тематичних розділів, кожен з яких передбачає вивчення теоретичного матеріалу та закріплення практичних навичок через написання запитів до тестової

бази даних. Передбачається, що користувач буде проходити розділи по черзі, щоб збільшувати складність поступово, проте можна обрати розділ з необхідно темою, і почати вивчення з нього. Цей тренажер має уніфікований інтерфейс, з яким доволі просто та зручно працювати, немає зайвих опцій тому можна швидко знайти необхідний функціонал. Під час написання запитів, що передбачають вибірку даних, користувачу навіть немає потреби відправляти запити. Перевірка та відображення результату для таких запитів відбувається динамічно, що значно зменшує витрати часу при написанні схожих за структурою запитів. Результат виконання запиту подається у вигляді таблиці, що допомагає краще зрозуміти представлення даних, з якими працює користувач.

Оригінальне рішення – «Web-додаток оцінювання лабораторних робіт з дисципліни “Організація баз даних та знань”» (db-sql-practics) [12]. Вигляд сторінки тренування навичок створення вибірки даних для цього web-додатоку наведено на рисунку 1.5.

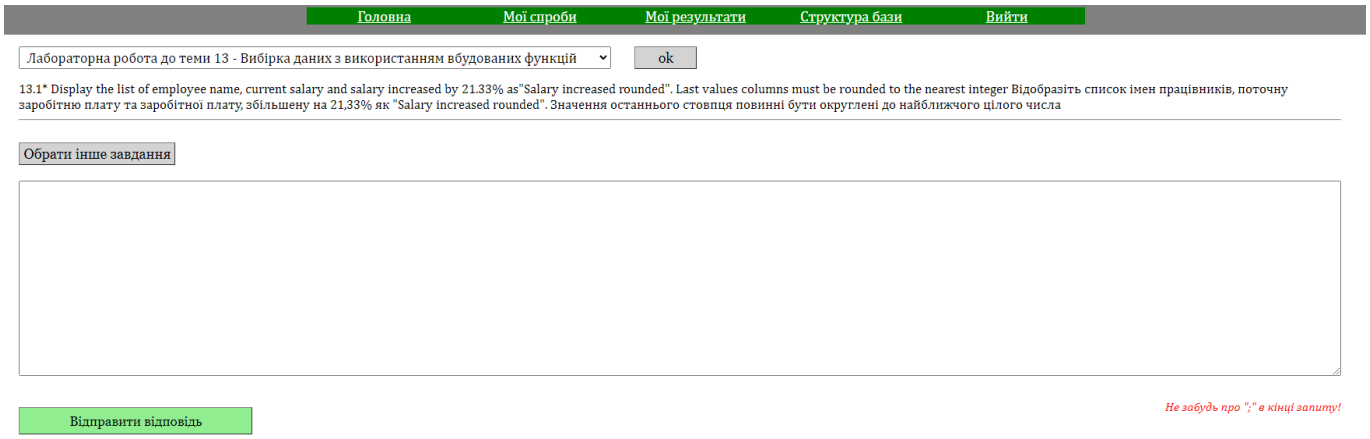


Рисунок 1.5 – Сторінка для тренування навичок створення вибірки даних SQLBolt

Даний web-додаток дозволяє авторизованими користувачам (студентам) виконувати завдання з лабораторних робіт, кожна з яких відповідає окремій темі, та отримувати оцінку за виконання цих завдань. Інтерфейс додатку – простий, але неадаптивний, естетично не дуже привабливий та дещо неприродний, тому ефективність під час роботи з додатком може бути середньою. Користувач має такий

набір опцій: виконання обраного завдання з лабораторної роботи, перегляд своїх спроб та результатів по кожному завданню, а також перегляд структури бази даних, з якою він працює. Перевірка правильності, наданих користувачем, відповідей працює не зовсім точно, та доволі часто вимагає від викладача власноруч передивлятися відповіді студентів. Відображення отриманого, після виконання запиту, результату не передбачає інформативного пояснення у разі виникнення помилки.

Складемо порівняльну таблицю за спільними та найбільш значущими властивостями розглянутих рішень (табл. 1.1). Введемо позначення: якщо певний ресурс має зазначену властивість, то він отримує позначку «+», інакше «-».

Таблиця 1.1 – Порівняння аналогів

	SQLCourse	SQLZoo	HakerRank	SQLBolt	db-sql-practics	db-sql-practics (new)
Зручний та ефективний інтерфейс	-	+	+	+	-	+
Інформативне сповіщення у разі виникнення помилки під час виконання запиту	-	+	+	+	-	+
Автоматизована адаптація синтаксису запиту	+	+	+	+	-	+

Продовження таблиці 1.1

	SQLCourse	SQLZoo	HakerRank	SQLBolt	db-sql-practics	db-sql-practics (new)
Динамічна перевірка правильності запиту	-	-	-	+	-	-
Висока точність оцінювання правильності наданої відповіді	-	+	+	+	-	+
Допоміжні опції для написання запитів	-	-	-	-	-	+

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі дослідження

Метою виконання роботи є модернізація архітектури програмного забезпечення ІС для дисципліни «Організація баз даних та знань», і вдосконалення програмного модуля автоматизованої перевірки завдань з лабораторних робіт. Оновлена ІС, не враховуючи вже наявного функціоналу, має вирішувати ряд задач:

- надання інтерфейсу для ефективної роботи з системою;
- автоматизоване визначення правильності, наданих студентами, відповідей до завдань;
- відображення пояснень рішення підсистеми перевірки завдань, щодо правильності/неправильності відповідей студентів;
- надання можливості фільтрації та сортування завдань і відповідей за визначеними критеріями.

Досягнення мети проекту передбачає виконання ряду завдань:

- дослідження та аналіз предметної області;
- виконання планування проекту;
- проведення зворотної розробки наявної системи;
- зміна архітектури додатку відповідно до обраного архітектурного шаблону;
- створення бази знань з правилами оцінки відповідей студентів;
- реалізація модуля з підсистемою перевірки завдань, що використовує синтаксичний аналізатор SQL запитів і правила бази знань;
- впровадження більш зручного користувальницького інтерфейсу;
- тестування функціональних модулів;
- створення загальної документації.

2.2 Засоби реалізації

Для більш зручного створення програмного забезпечення ІС використовується редактор коду Visual Studio Code, що є дуже зручним інструментом для розробки та відлагодження web-застосунків. Даний редактор дозволяє використовувати спеціальні надбудови, що спрощують рефакторинг та аналіз програмного коду.

Розробка застосунку відбувається локально, тому для реалізації сереверної частини використовується Open Server Panel (OSPanel), що є портативним програмним середовишем, яке надає локальний web-сервер та ряд інструментів для розробки серверного ПЗ. Важливим інструментом OSPanel, є phpMyAdmin, що використовується для адміністрування MySQL через інтернет, і дозволяє проводити операції з базою даних як за допомогою інтерфейсу, так і через текстовий редактор для SQL запитів. Відповідно до функціональних вимог сереверної частини було вирішено використовувати систему керування базами даних MySQL версії 8.0, а також мову програмування PHP версії 7.2.

Клієнтська частина передбачає розробку користувальницького інтерфейсу, який відборажатиметься в обраному користувачем браузері. Інтерфейс має бути універсальним та адаптивним, тому для створення структури web-сторінок використовується мова розмітки HTML5, для стилізації цих сторінок – таблиці стилів CSS3, та фреймвор Bootstrap версії 5.1, який надає дуже велику кількість структурних та стилістичних шаблонів для побудови сторінок. Для того, щоб інтерфейс був інтерактивними, а також для додавання функціоналу, що полешить та пришвидшить роботу з таблицями даних шляхом використання скриптиів до цих структурних елементів використовується мова програмування JavaScript на основі ESMAScript 8.

3 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ АВТОМАТИЗОВАНОЇ ПЕРЕВІРКИ ЛАБОРАТОРНИХ РОБІТ З ВИБІРКИ ДАНИХ

3.1 Структурно функціональне моделювання процесу «Організація виконання та оцінювання завдань з лабораторних робіт»

У даному підрозділі наводить модель системи в нотації IDEF0. Вона включає в себе контекстну діаграму та діаграми декомпозиції першого і другого рівнів. На вхід системи поступає:

- Номер лабораторної роботи, що визначає необхідну до виконання лабораторну роботу і набір завдань, що відповідають цій роботі.

Елементами керування є:

- Завдання з лабораторних робіт, що містять опис того, яку інформацію повинен отримати студент в результаті виконання запиту;
- Правила бази знань, на основі яких проводиться співставлення синтаксичних лексеми з відповідей студентів та правильних відповідей;
- Алгоритм синтаксичного аналізу, що необхідний для розкладання відповіді студента (SQL запиту) на набір лексем;
- Правильні відповіді, що використовуються для автоматизованого оцінювання SQL запитів та підтвердження правильності цього оцінювання викладачем.

Механізмами виступають:

- Студент, який виконує завдання з лабораторної роботи;
- Викладач, який має змогу підтвердити або відкоригувати попередню оцінку;
- Модуль пошуку та виконання завдань, який використовується для вибору необхідної лабораторної роботи і завдання до неї, а також формування відповіді на цих завдання;

- Модуль автоматизованого оцінювання робіт, в якому виконується синтаксичний аналіз, отриманих від студентів, відповідей та встановлення попередньої оцінки на основі правил бази знань.

На виході системи отримуємо:

- Оцінку за виконану лабораторну роботу, що складається з суми оцінок за кожне завдання цієї роботи.

Контекстна діаграма, що представляє основний процес – «Організація виконання та оцінювання завдань з лабораторних робіт», наведена на рисунку 3.1.

Рисунок 3.1 – Контекстна діаграма

На рисунку 3.2 наведена діаграма декомпозиції 1-го рівня для процесу «Організація виконання та оцінювання завдань з лабораторних робіт» (A0), що розкладається на два процеси: «Виконання лабораторної роботи» (A1), «Перевірка та оцінювання наданих відповідей» (A2).

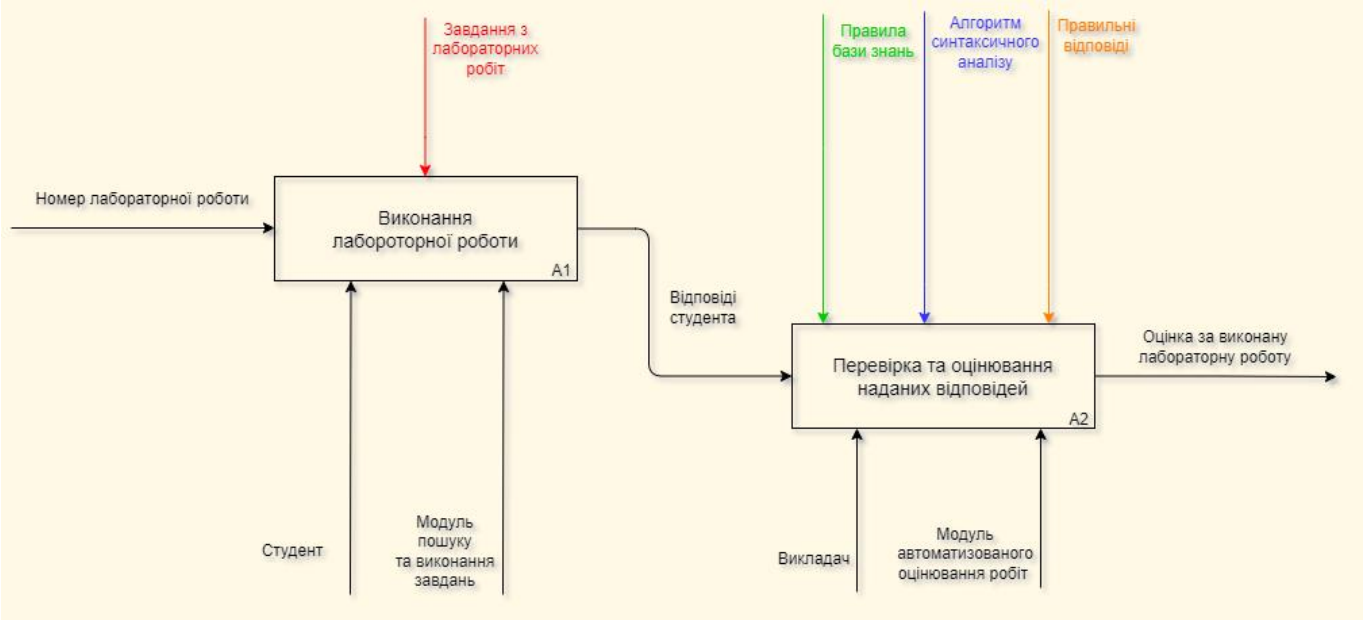


Рисунок 3.2 – Діаграма декомпозиції 1-го рівня

На риунку 3.3 зображено діаграму декомпозиці 2-го рівня для процесу «Перевірка та оцінювання наданих відповідей» (A2), що розкладається на 4 послідовних процеси: «Приведення SQL запитів до загального формату» (A2.1), «Синтаксичний аналіз SQL запитів» (A2.2), «Оцінювання правильності SQL запитів» (A2.3) і «Підтвердження результатів оцінювання» (A2.4).

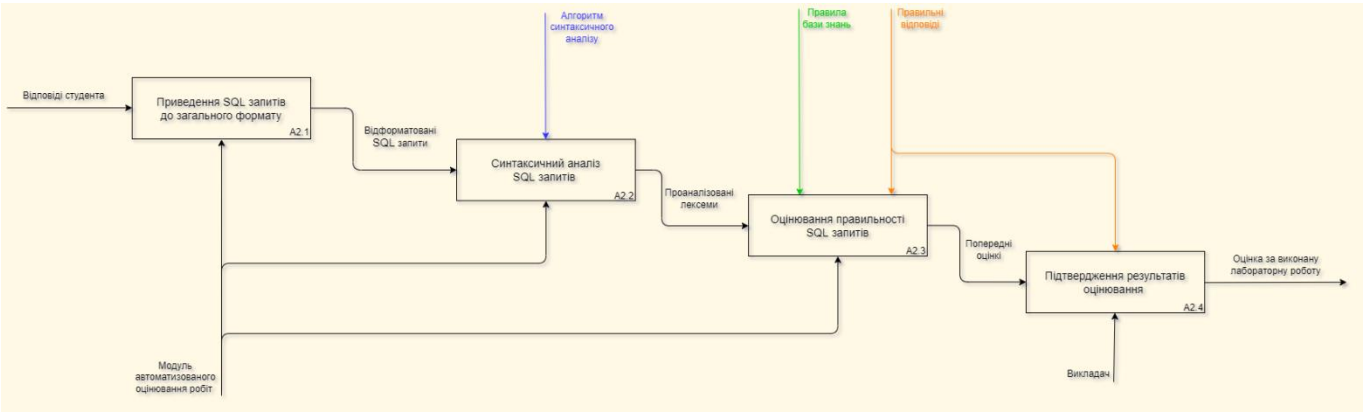


Рисунок 3.3 – Діаграма декомпозиції 2-го рівня для процесу «Перевірка та оцінювання наданих відповідей»

3.2 Проектування моделі бази даних

У даному підрозділі наводяться концептуальна і даталогічна моделі бази даних системи. На рисунку 3.4 наведено концептуальну модель бази даних, на якій відображені відношення між сутностями та властивості цих сутностей.

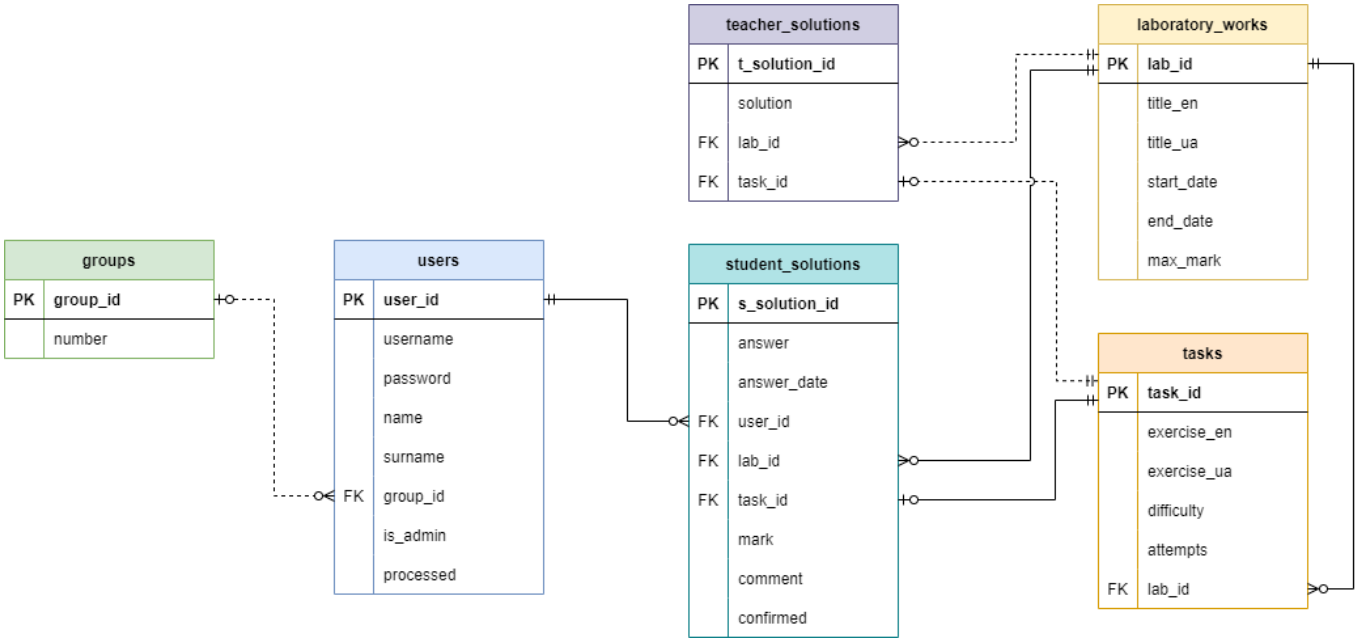


Рисунок 3.4 – Концептуальна модель бази даних системи

База даних системи містить такі сутності:

- groups: містить номер групи, до якої можуть належати студенти;
- users: містить інформацію про користувачів системи;
- laboratory_works: містить інформацію про лабораторні роботи;
- tasks: містить інформацію про завдання з лабораторних робіт;
- teacher_solutions: містить, встановлені викладачем, правильні відповіді до завдань з лабораторних робіт;
- student_solutions: містить, надіслані студентами, відповіді.

Передбачений ряд наступних зв’язків між сутностями:

- groups - users: група може містити багато користувачів / користувач може належати до однієї групи, при цьому користувач може існувати без групи;
- users - student_solutions: користувач може надсилати багато відповідей / надіслана відповідь має належати одному користувачу;
- laboratory_works - tasks: лабораторна робота може містити багато завдань / завдання має належати лише до однієї лабораторної роботи;
- laboratory_works - teacher_solutions: лабораторній роботі можуть належати багато правильних відповідей на завдання / правильна відповідь повинна відноситись лише до однієї лабораторної роботи, при цьому правильна відповідь може існувати без лабораторної роботи;
- laboratory_works - student_solutions: лабораторній роботі можуть належати багато, надісланих студентами, відповідей на завдання / надіслана студентом, відповідь повинна відноситись лише до однієї лабораторної роботи, ;
- tasks - teacher_solutions: завдання може містити лише одну правильну відповідь / правильна відповідь має відноситись лише до одного завдання, при цьому правильна відповідь може існувати без завдання;
- tasks - student_solutions: завдання може містити лише одну, надіслану студентом, відповідь / надіслана студентом, відповідь має відноситись лише до одного завдання.

На рисунку 3.5 наведено даталогічну модель бази даних, яка, окрім зв’язків між сутностями, містить типи даних для кожної з властивостей.

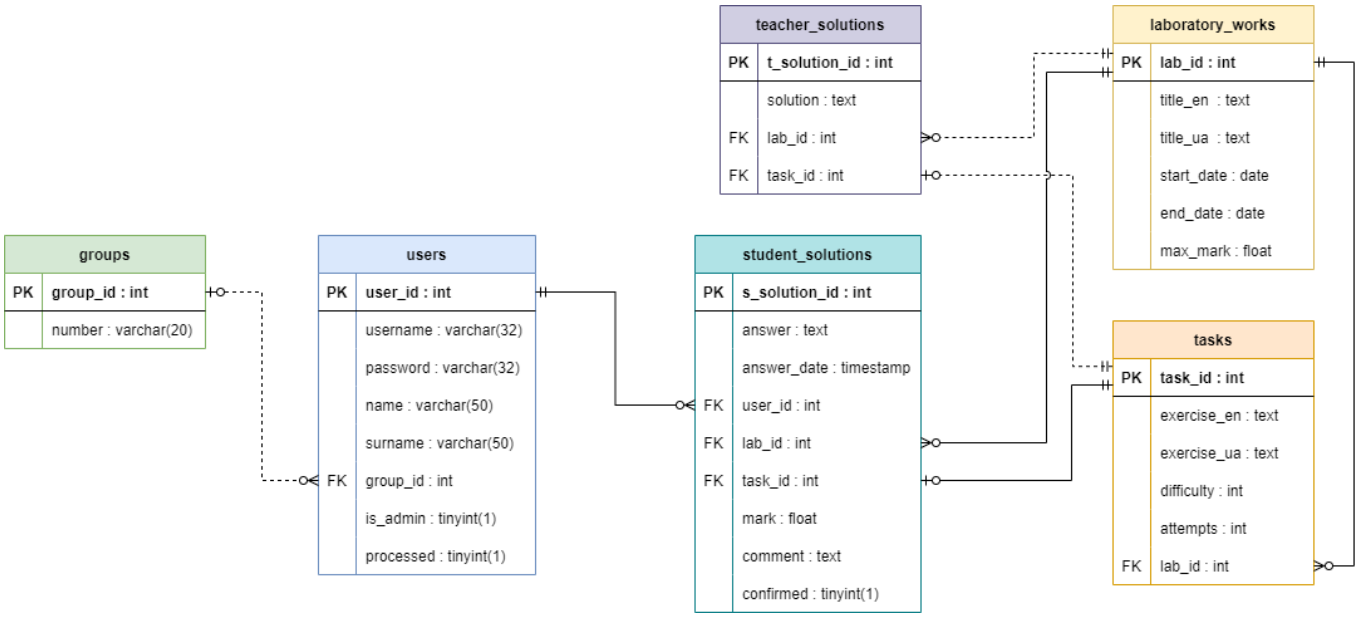


Рисунок 3.5 – Даталогічна модель бази даних системи

Для перевірки синтаксичної правильності відповіді, наданої студентом, використовується навчальна база даних. Концептуальна модель навчальної бази даних представлена на рисунку 3.6.

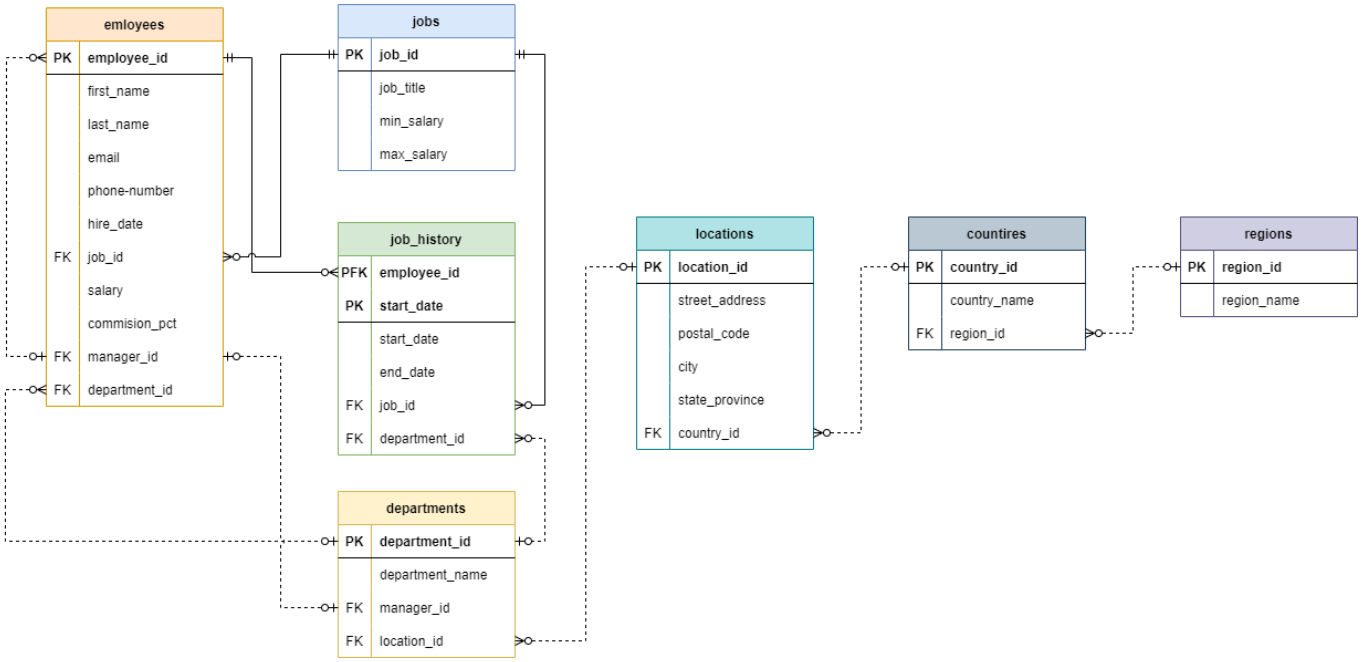


Рисунок 3.6 – Концептуальна модель навчальної бази даних

Навчальна база даних містить такі сутності:

- employees: містить інформацію про співробітників, кожному з яких призначається посада та, можливо, керівник;
- jobs: містить інформацію про посади в компанії;
- job_history: містить інформацію про поточну та попередні посади співробітника у певному відділі;
- departments: містить інформацію про відділі компанії;
- regions: містить географічні області;
- countries: містить інформацію про регіони для кожної країни
- locations: містить адреси, які можуть відповідати певній країні.

Для перевірки семантичної правильності відповіді, наданої студентом, використовується база знань. Концептуальна модель бази даних для бази знань представлена на рисунку 3.7.

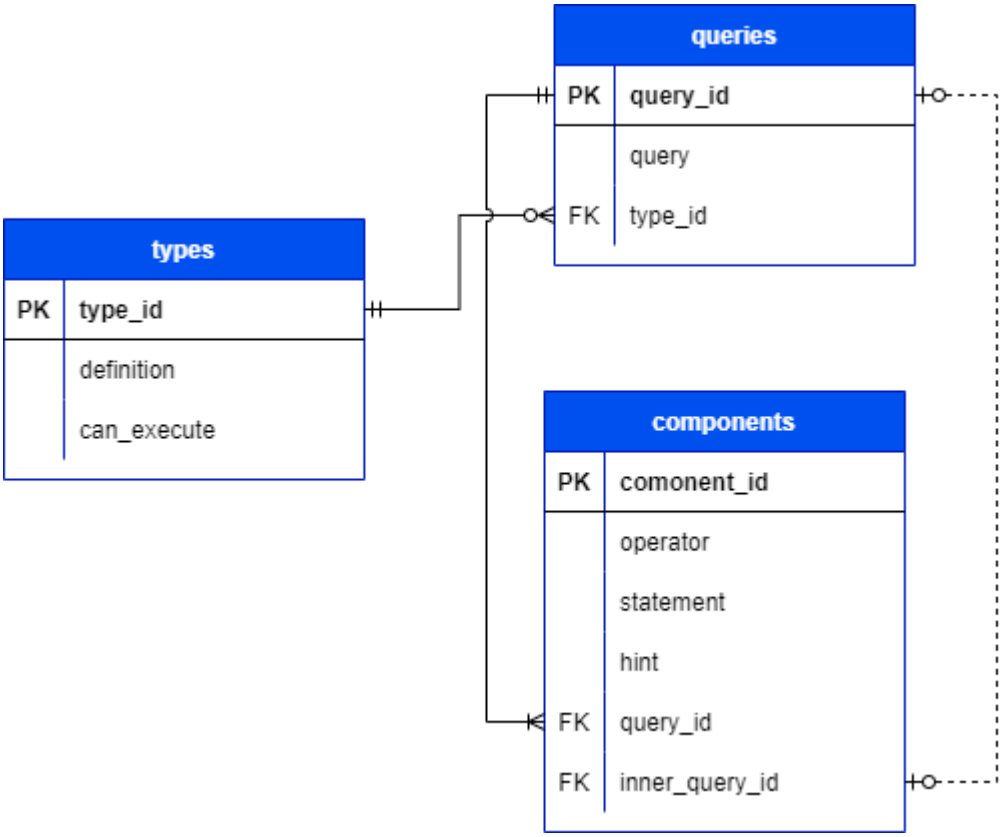


Рисунок 3.7 – Концептуальна модель бази даних для бази знань

Дана модель складається з трьох, пов’язаних між собою, сутностей:

- queries: представляє собою правильну відповідь до певного завдання (відповідає сутності teacher_solutions із моделі бази даних ІС);
- types: містить визначення типу SQL запиту та можливість його виконання на навчальній базі даних;
- components: представляє собою лексему запиту, яка складається з SQL оператора і виразу, що розташований після нього, але лише до наступного SQL оператора; крім цього, містить підказку, яка відображатиметься у разі неспівпадіння компоненту з відповіді студента, до відповідного йому компонента визначеної правильної відповіді; також, може містити ідентифікатор внутрішнього запиту, якщо правильна відповідь включає в себе підзапит.

3.3 Моделювання варіантів використання

У даному підрозділі наводиться діаграма варіантів використання системи (Use Case Diagram). Крім того, надається короткий опис прецедентів, акторів і артефактів системи. Визначено таких акторів (4):

- Викладач: користувач, що виступає адміністратором системи, він виконує CRUD-операції над лабораторними роботами, завданнями до них та списком користувачів системи, може переглядати та підтверджувати відповіді студентів, а також формувати звіт з оцінок студентів за допомогою MS Excel;
- Студент: користувач, що шукає та виконує завдання з лабораторних робіт, може переглядати свої оцінки до виконаних завдань;
- MySQL: система керування базою даних, що використовується для всіх варіантів використання, що передбачають роботу із записами бази даних;
- База знань: набір правил для оцінювання відповідей студентів.

Наступні варіанти використання (14):

- Авторизація: визначення ролі користувача у системи та надання доступу до відповідного функціоналу;
- Реєстрація: надання можливості авторизації у системі;
- Створення звіту MS Excel: формування документу MS Excel, що містить оцінки студентів за лабораторні роботи;
- Перегляд наявних відповідей: перегляд відповідей, надісланих студентами;
- Підтвердження оцінок до відповідей: підтвердження або корегування оцінок, попередньо встановлених автоматизованим оцінюванням відповідей;
- Перегляд структури навчальної бази;
- CRUD-операції з лабораторними роботами: створення, відображення, редагування або видалення лабораторних робіт з бази даних;
- CRUD-операції із завданнями: створення, відображення, редагування або видалення завдань з лабораторних робіт з бази даних;
- CRUD-операції зі списком користувачів: створення, відображення, редагування або видалення користувачів з бази даних;
- Перегляд оцінок за виконані завдання;
- Пошук завдань з лабораторних робіт;
- Виконання завдань з лабораторних робіт;
- Оцінка правильності відповіді: надання попередньої оцінки відповіді студента до певного завдання лабораторної роботи;
- Синтаксичний аналіз відповідей: розкладання відповідей (SQL запитів) на лексеми для оцінювання кожного з фрагментів запиту, та встановлення пояснень у разі хибності відповіді.

Такі артефакти (3):

- Інформація користувачів: дані для ідентифікації користувачів (ім’я, прізвище, група);
- Завдання з лабораторних робіт: опис та пояснення того, що повинен виконувати запит користувача у відповідному завданні;
- Правильні відповіді: заздалегідь визначені та відформатовані відповіді, що використовуються для співставлення з відповідями студентів.

Діаграма варіантів використання наведена на рисунку 3.8.

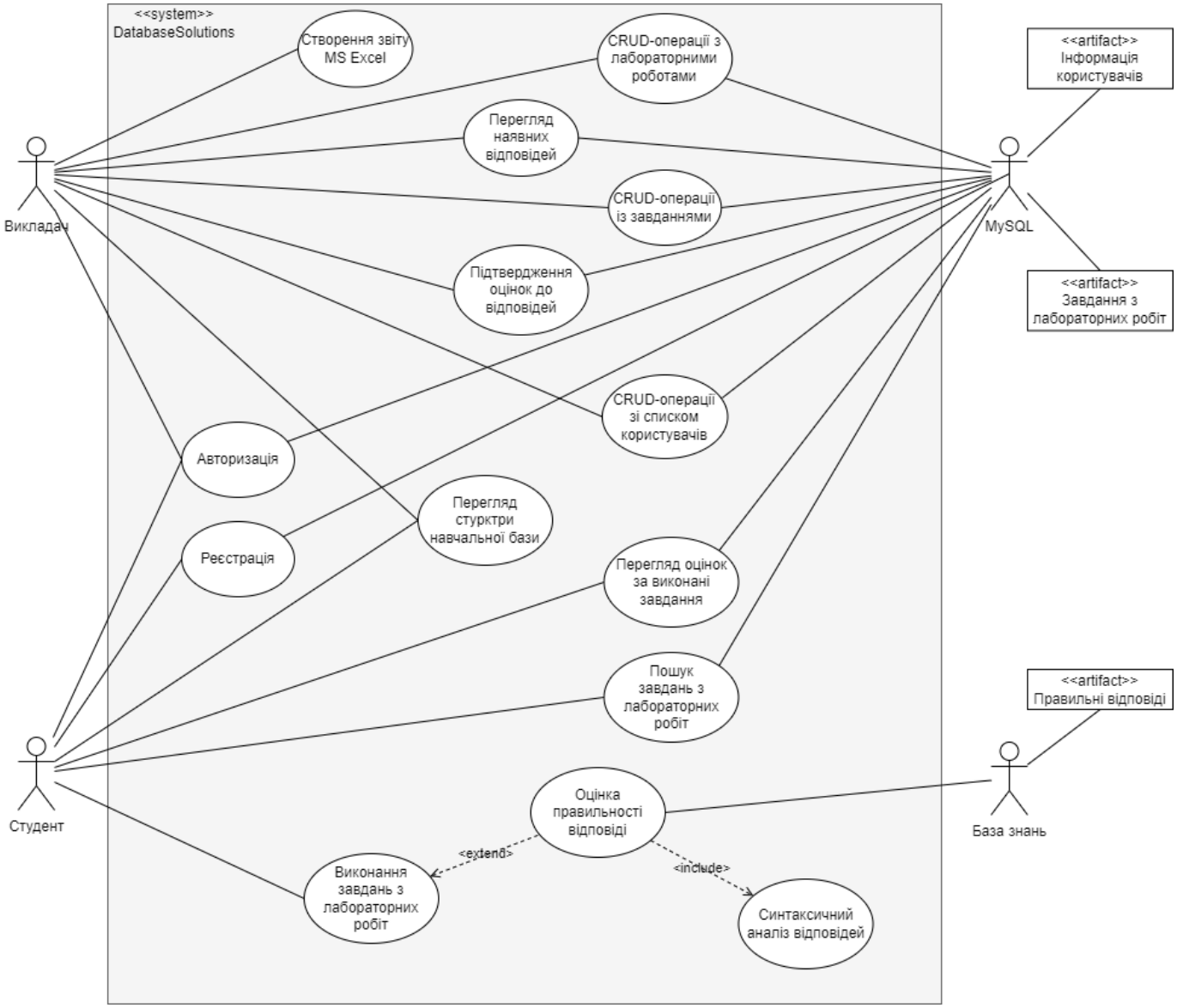


Рисунок 3.8 – Діаграма варіантів використання

4 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ АВТОМАТИЗОВАНОЇ ПЕРЕВІРКИ ЛАБОРАТОРНИХ РОБІТ З ВИБІРКИ ДАНИХ

4.1 Архітектура програмного забезпечення

В якості архітектурного шаблону для розробки програмного забезпечення ІС було обрано Model-View-Controller (MVC), що використовується для побудови більшості web-орієнтованих ІС. MVC складається з трьох основних компонентів: моделі, що відповідає за бізнес-логіку застосунку; вигляду, за допомогою якого формується користувальницький інтерфейс; контролера, який забезпечує взаємозв’язок між виглядом та моделлю.

Користувач робить певний запит до системи, який потрапляє у контролер. В свою чергу, контролер визначає певну модель або моделі, які буду відповідати за обробку запиту, а також, вигляд, який застосовуватиметься для відображення результату. Потім, контролер передає запит користувача на обробку до визначеної моделі. Функціонування моделі передбачає обмін та обробку даних, отриманих з бази даних та бази знань. База даних обмінюється з моделлю інформацію про лабораторні роботи та користувачів системи, а база знань – інформацією для оцінки правильності відповідей студентів. Виконавши запит користувача, модель повертає результат запиту до контролеру. Тепер, контролер, передає необхідні дані, отримані з моделі, у вигляд для їх стилізованого та відформатованого відображення користувачу, і таким чином, користувач отримує відповідь на свій запит.

Архітектура програмного забезпечення ІС наведена на рисунку 4.1.

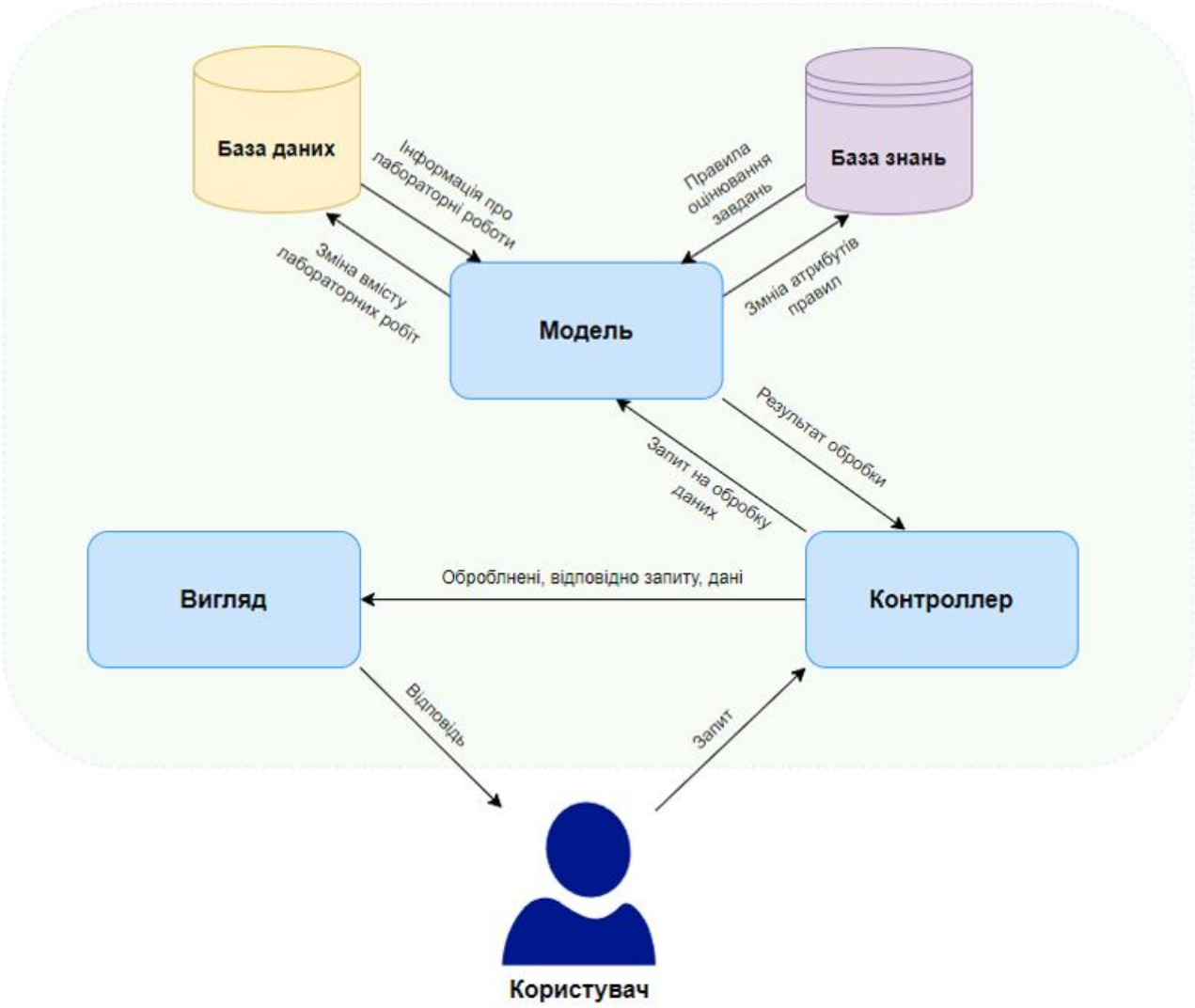


Рисунок 4.1 – Архітектура програмного забезпечення ІС

4.2 Програмна реалізація

Для кожного з основних архітектурних компонентів додатку було створено відповідні директорії, в яких розташовуються більш специфічні фрагменти моделі, контролера та вигляду. Файлова структура проекту наведена на рисунку 4.2.

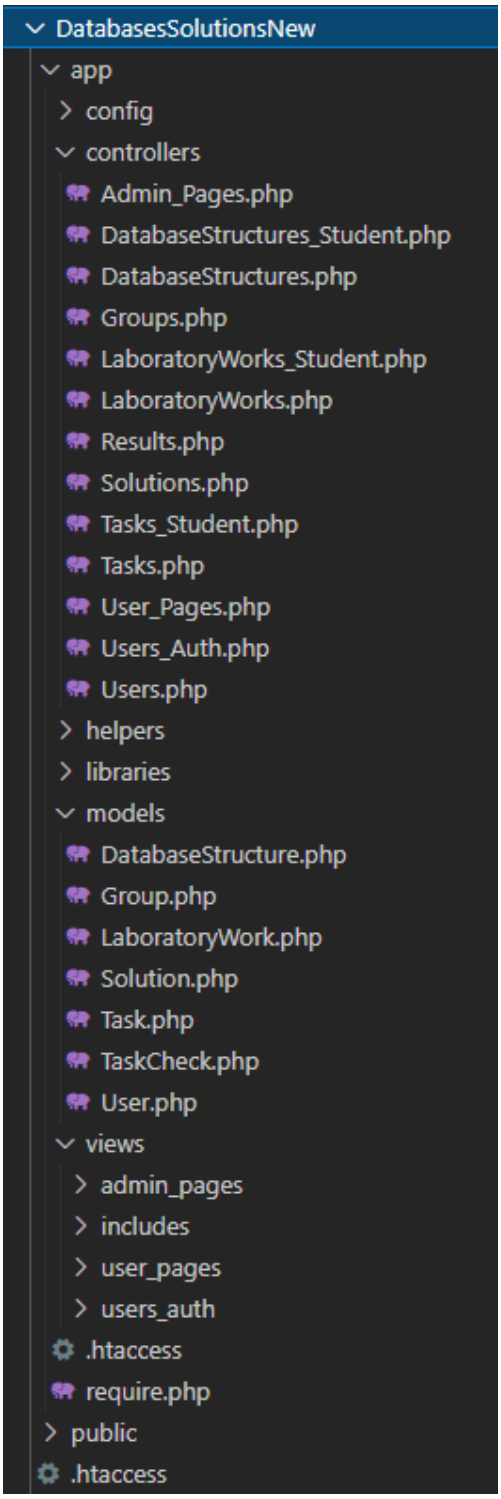


Рисунок 4.2 – Файлова структура проекту

Для визначення параметрів підключення до основної та навчальної баз даних, а також ряду параметрів для навігації було створено окремий файлу – *config.php*, в якому ці параметри встановлюються як константні значення (рис. 4.3).

```php
<?php
    // Database connection params.
    define('DB_HOST', 'localhost');
    define('DB_NAME', 'databasesolutions');
    define('DB_USER', 'mysql');
    define('DB_PASS', 'mysql');
    // Educational Database connection params.
    define('DB_HOST_EDC', 'localhost');
    define('DB_NAME_EDC', 'educational');
    define('DB_USER_EDC', 'mysql');
    define('DB_PASS_EDC', 'mysql');
    // Root to the app folder.
    define('APPROOT', dirname(dirname(__FILE__)));
    // Main URL.
    define('URLROOT', 'http://databasessolutionsnew');
    // Site alias.
    define('SITEALIAS', 'Databases Solutions');
```

Рисунок 4.3 – Лістинг файлу *config.php*

Основним класом, який керує роботою додатку, є клас *Core* (рис. 4.4). В ньому, за допомогою методу *getUrl()* відбувається парсинг адресного рядка та створення масиву з компонентів цього рядка. Перший елемент масиву містить назву контролера, який буде використовуватись. Після ініціалізації класу контролера відбувається визначення методу цього коньролера, що має викликатися. Для кожного, починаючи з другого, елементів масиву відбувається перевірка на те, чи інснує метод з ім’ям, що містить елемент масиву, в ініціалізованому контролері. Якщо певний елемент масиву не проходить перевірку, то він видаляється з масиву, а пошук продовжується. Якщо метод з існуючим ім’ям знайдено, то пошук завершується, а всі елементи масиву, що залишились після поточного, стають параметрами методу. В результаті відбувається виклик визначенго методу зі встановленого контролера, з параметрами або без них. Для випадку, коли контролер або метод не вдається визначити, відбувається ініціалізація контролара авторизації користувачів (*User_Auth*) та виконання методу авторизації (*login*), тобто, переадресація на початкову сторінку системи.

```php
<?php
    class Core {
        protected $currentController = 'Users_Auth';
        protected $currentMethod = 'login';
        protected $params = [];

        public function __construct() {
            $url = $this->getUrl();
            // Getting controller name as a first array element.
            if($url)
            {
                $controller = ucwords($url[0], "_");
                if(file_exists('../app/controllers/' . $controller . '.php')) {
                    $this->currentController = $controller;
                    unset($url[0]);
                }
            }

            require '../app/controllers/' . $this->currentController . '.php';
            $this->currentController = new $this->currentController;
            // Getting method name by iterating over remaining array elements and searching for existing name.
            // Strict naming rules for files and controller`s methods!
            // Array elements remaining after searching for method become parameters of the method.
            if($url)
            {
                $urlCount = count($url);
                for($i = 1; $i <= $urlCount; $i++) {
                    $method = $url[$i];
                    if(method_exists($this->currentController, $method)) {
                        $this->currentMethod = $method;
                        unset($url[$i]);
                        break;
                    }
                    unset($url[$i]);
                }

                $this->params = array_values($url);
            }
            call_user_func_array([$this->currentController, $this->currentMethod], $this->params);
        }

        public function getUrl() {
            if(isset($_GET['url'])) {
                $url = rtrim($_GET['url'], '/');
                $url = filter_var($url, FILTER_SANITIZE_URL);
                $url = explode('/', $url);
                return $url;
            }
        }
    }
```

Рисунок 4.4 – Лістинг файлу *Core.php*

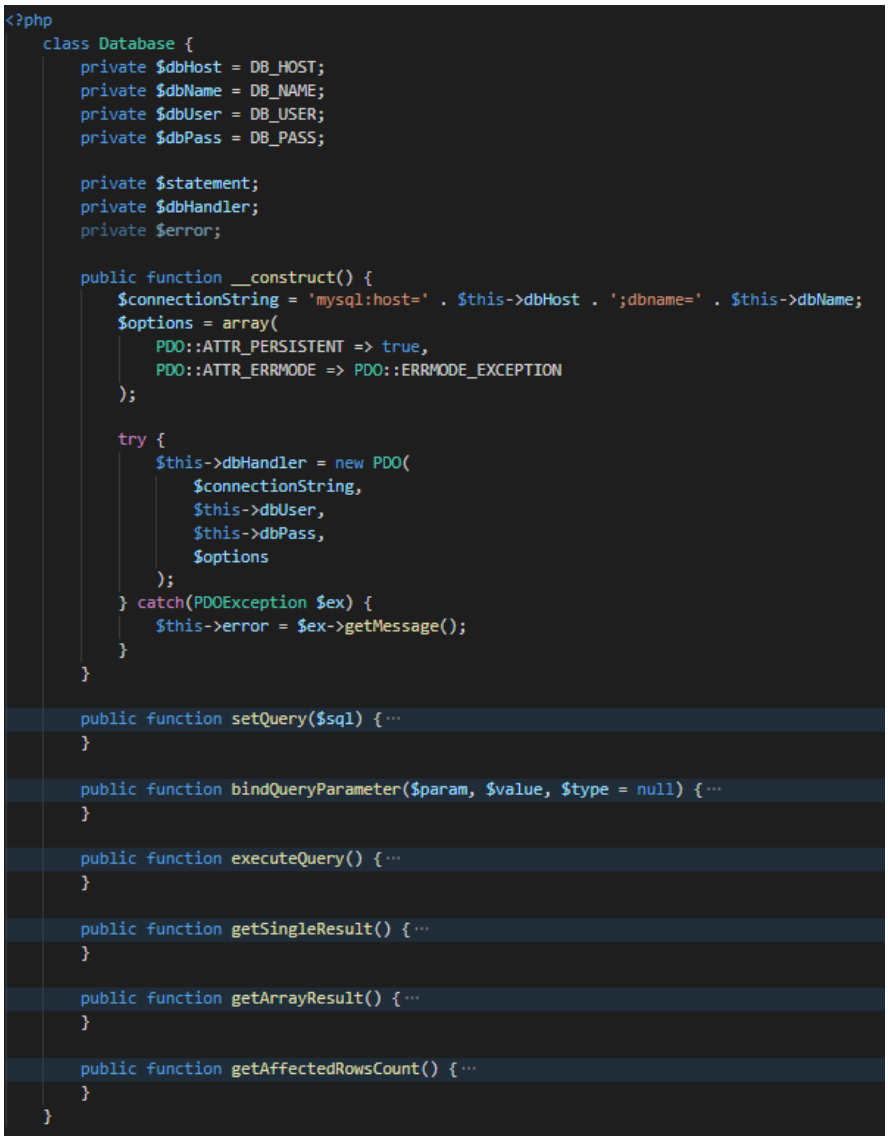
Головний (батьківський) клас контролера містить лише два методи: *model($model)* – для ініціалізації моделей та *view($view, data=[])* – для ініціалізації виглядів, що окрім параметру вигляду, містить необов’язковий параметр для передачі оброблених даних з контролера у вигляд (рис. 4.5).

```php
<?php
    class Controller {
        public function model($model) {
            require '../app/models/' . $model . '.php';
            return new $model();
        }

        public function view($view, $data = []) {
            if(file_exists('../app/views/' . $view . '.php')) {
                require '../app/views/' . $view . '.php';
            } else {
                die("No view.");
            }
        }
    }
```

Рисунок 4.5 – Лістинг файлу *Controller.php*

Для організації доступу до бази даних системи використовується клас *Database*. У конструкторі цього класу формується рядок з’єднання та встановлюється з’єднання з базою даних за допомогою створення нового об’єкту даних PHP (PDO), який визначає інтерфейс доступу до баз даних MySQL (рис. 4.6). Також, клас містить ряд методів, необхідних для виконання запитів та отримання результатів від бази даних: *setQuery($sql)* – визначає вміст SQL запиту, *bindQueryParameter(…)* – визначає тип параметра та встановлює його для відповідного значення у сформованому запиті, *executeQuery()* – проводить виконання сформованого запиту, *getSingleResult()* *getArrayResult()* та *getAffectedRowsCount()* – повертають єдиний рядок, масив рядків та кількість рядків результату, відповідно.



```php
<?php
    class Database {
        private $dbHost = DB_HOST;
        private $dbName = DB_NAME;
        private $dbUser = DB_USER;
        private $dbPass = DB_PASS;

        private $statement;
        private $dbHandler;
        private $error;

        public function __construct() {
            $connectionString = 'mysql:host=' . $this->dbHost . ';dbname=' . $this->dbName;
            $options = array(
                PDO::ATTR_PERSISTENT => true,
                PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION
            );

            try {
                $this->dbHandler = new PDO(
                    $connectionString,
                    $this->dbUser,
                    $this->dbPass,
                    $options
                );
            } catch(PDOException $ex) {
                $this->error = $ex->getMessage();
            }
        }

        public function setQuery($sql) {···
        }

        public function bindQueryParameter($param, $value, $type = null) {···
        }

        public function executeQuery() {···
        }

        public function getSingleResult() {···
        }

        public function getArrayResult() {···
        }

        public function getAffectedRowsCount() {···
        }
    }
```

Рисунок 4.6 – Лістинг файлу *Database.php*

Розглянемо принцип реалізований принцип роботи контролера на прикладі контролера авторизації/реєстрації користувачів (*User_Auth.php*) (рис. 4.7). Кожен з класів контролерів, що відповідає певній сторінці, успадковується від головного контролера. Таким чином, в середині конструкторів цих класів відбувається ініціалізація необхідних моделей (у даному випадку – *User* та *Group*), що необхідні для обробки та надання даних, запит на які робить користувач. Метод *login()*, який викликається при завантаженні системи передбачає лише завантаження вигляду з формою авторизації, при цьому масив параметрів *$data*, який передається у вигляд є фактично пустим. Після того, як користувач введе певні значення у поля форми авторизації та натисне кнопку для відправки форми контролер знову викличне метод

login(), проте тепер, масив *$data* буде заповнюватись значеннями, які ввів користувач у відповідні поля. У випадку, якщо значення певного поля не пройшло валідацію, то в масиві *$data* встановлюються параметри з повідомленням про помилку, які будуть надіслані у поточний вигляд. Якщо помилок введені дані безпомилкові, то викликається відповідний метод моделі, для обробки запиту на авторизацію. Далі, якщо обробку запиту виконано успішно – створюється сесія користувача, та в залежності від ролі користувача в системі, відбувається переадресація на головну сторінку викладача або студента.

```php
<?php
    class Users_Auth extends Controller {
        private $userModel;
        private $groupModel;

        public function __construct() {
            $this->userModel = $this->model('User');
            $this->groupModel = $this->model('Group');
        }

        private function createSession($user) {···
        }

        public function login() {
            $data = [
                'username' => '',
                'password' => '',
                'usernameError' => '',
                'passwordError' => ''
            ];

            if($_SERVER['REQUEST_METHOD'] == 'POST') {
                $_POST = filter_input_array(INPUT_POST, FILTER_SANITIZE_STRING);
                $data = [
                    'username' => trim($_POST['username']),
                    'password' => trim($_POST['password']),
                    'usernameError' => '',
                    'passwordError' => ''
                ];

                if(!$data['username']) {
                    $data['usernameError'] = 'Enter the username, please.';
                }

                if(!$data['password']) {
                    $data['passwordError'] = 'Enter the password, please.';
                }

                if(!$data['usernameError'] && !$data['passwordError']) {
                    $loggedInUser = $this->userModel->login(array_slice($data, 0, 2));

                    if($loggedInUser == NOT_PROCESSED) {
                        $data['passwordError'] = 'You need to wait for confirmation of registration from the teacher.';
                    }
                    if($loggedInUser == INCORRECT_CREDENTIALS) {
                        $data['passwordError'] = 'Username or password is incorrect. Please try again.';
                    }
                    if(!$data['passwordError']) {
                        $this->createSession($loggedInUser);
                        if(isAdmin()) {
                            header('location:' . URLROOT . '/admin_pages/labs_tasks/laboratory_works');
                        } else {
                            header('location:' . URLROOT . '/user_pages/main');
                        }
                    }
                }
            }

            $this->view('users_auth/login', $data);
        }
```

Рисунок 4.7 – Фрагмент лістингу файлу контролера *User_Auth.php*

Для обробки запиту на авторизацію та ряду інших операцій із інформацією про користувачів використовується клас моделі *User* (рис. 4.8). Коли контролер викликає метод моделі *login($data)* в нього передаються введені користувачем значення логіна та пароля. Далі, формується запит на знаходження користувача з цими параметрами, цей запит виконується, а результат виконання перевіряється на відповідність значення пароля та наявність прав на авторизацію, що надає викладач. У разі невдалого проходження перевірки, метод повертає код помилки, а при успішній перевірці – дані користувача.

```php
<?php
    class User {
        private $db;

        public function __construct() {
            $this->db = new Database;
        }

        public function getAdmins() {···
        }

        public function getStudentsWithGroupOnProcessed($processed) {···
        }

        public function getUserById($id) {···
        }

        public function login($data) {
            $this->db->setQuery('SELECT *
                                 FROM `users`
                                 WHERE username = :username');

            $this->db->bindQueryParameter(':username', $data['username']);

            $resultRow = $this->db->getSingleResult();
            $hashedPassword = $resultRow->password;
            $is_processed = boolval($resultRow->processed);
            if(password_verify($data['password'], $hashedPassword)) {
                if($is_processed) {
                    return $resultRow;
                } else {
                    return NOT_PROCESSED;
                }
            } else {
                return INCORRECT_CREDENTIALS;
            }
        }

        public function register($data) {···
        }
```

Рисунок 4.8 – Фрагмент лістингу файлу моделі *User.php*

Результат обробки даних з моделі повертається в контролер, який виконує повторну ініціалізацію вигляду з формою авторизації (у випадку, якщо авторизація

не пройшла успішно), в яку передаються відповідні параметри, що відображаються користувачеві (рис. 4.9).

```php
<?php require APPROOT . '/views/includes/header.php'; ?>
<?php require APPROOT . '/views/includes/common_res.php'; ?>

<div class="container">
    <div class="row h-100 align-items-center">
        <div class="col-4 offset-4">
            <div class="d-flex justify-content-center">
            <h2>Sign in</h2>
            </div>
            <form action="<?= URLROOT; ?>/users_auth/login" method="POST">
                <div class="form-group">
                    <div class="col-8 offset-2 mb-2">
                        <label for="username">Username</label>
                        <input type="text" class="form-control <?php if($data['usernameError']) { ?>is-invalid<?php } ?>"
                               id="username" name="username" placeholder="Enter username"
                               value="<?php echo ($_POST['username'] ? trim($_POST['username']) : $data['username']); ?>">
                        <div class="invalid-feedback"><?= $data['usernameError']; ?></div>
                    </div>
                </div>
                <div class="form-group mb-4">
                    <div class="col-8 offset-2">
                        <label for="password">Password</label>
                        <input type="password" class="form-control <?php if($data['passwordError']) { ?>is-invalid<?php } ?>"
                               id="password" name="password" placeholder="Password"
                               value="<?php echo ($_POST['pasword'] ? trim($_POST['password']) : $data['password']); ?>">
                        <div class="invalid-feedback"><?= $data['passwordError']; ?></div>
                    </div>
                </div>
                <div class="col-8 offset-2 mb-2">
                    <button type="submit" class="btn btn-primary col-12">Sign in</button>
                </div>
                <div class="col-8 offset-2">
                    <p class="text-center">First time here?<br/><a href="<?= URLROOT ?>/users_auth/register">Let`s register!</a></p>
                </div>
            </form>
        </div>
    </div>
</div>

<?php require APPROOT . '/views/includes/footer.php'; ?>
```

Рисунок 4.9 – Лістинг файлу вигляду *login.php*

Для автоматизованої перевірки відповіді до завдання, надісланої студентом, використовується метод *lab_task($lab_id, $task_id)* контролера *Tasks_Student*, параметрам якого є ідентифікатор певної лабораторної роботи та ідентифікатор завдання з цієї роботи (рис. 4.10). На початку методу відбувається запит до моделі для отримання даних про завдання, ідентифікатори наступного та попереднього завдань, правильної відповіді до поточного завдання, та кількості спроб, що залишились у студента. Якщо студень вперше за сесію роботи з системою відкрив цю сторінку, то відбувається завантаження вигляду із зазначеними вище даними.

Після введення відповіді до завдання у відповідне поле та натискання кнопки для підтвердження відповіді, відбувається двоетапна перевірка цієї відповіді та встановлення потенційної оцінки за виконання завдання. Спершу, відповідь

перевіряється на наявність синтаксичних помилок (якщо це SQL запит). При виявленні помилок на цьому етапі, студент отримує мінімальну оцінку, а у вигляд передається повідомлення з поясненням синтаксичної помилки. Якщо перевірка синтаксису пройшла успішно, то надана відповідь перевіряється на семантичну правильність відповідно до встановленої правильної відповіді. При виявленні помилок на цьому етапі, студент отримує знижену оцінку, що залежить від кількості помилок, а у вигляд передається повідомлення з поясненням семантичної помилки. І тільки у випаду, коли жодна з перевірок не повернула хибний результат, студент отримує максимальний бал за виконання завдання.

```php
public function lab_task($lab_id, $task_id) {
    $task = $this->taskModel->getTaskById($task_id);
    $prevNextIds = $this->taskModel->getNextAndPreviousTasksIds($lab_id, $task_id);
    $solutions = $this->solutionsModel->getStudentAttemptSolutions($task_id, $_SESSION['user_id']);
    $solutionsCount = $this->solutionsModel->getStudentAttemptsOnTaskCount($task_id, $_SESSION['user_id']);
    $attemptsLeft = $task->attempts - (int)$solutionsCount->solutions_count;

    $data = [···
    ];

    if($_SERVER['REQUEST_METHOD'] == 'POST') {
        $_POST = filter_input_array(INPUT_POST, FILTER_SANITIZE_STRING);

        $data = [···
        ];

        /* Check answer start. */

        $mark = $task->difficulty;
        $comment = "Succes.";
        $correctSolution = $this->solutionsModel->getCorrectSolutionByTaskId($task_id);
        $querySyntaxCheckRes = $this->taskCheckModel->checkQuerySyntax($data['answer']);
        if($querySyntaxCheckRes === true) {
            $querySemanticsCheckRes = $this->taskCheckModel->checkQuerySemantics($data['answer'], $correctSolution->t_solution_id);
            if($querySemanticsCheckRes === true) {
                $data['isAnswerCorrect'] = true;
            } else {
                $data['errorDetail'] = implode(" ", $querySemanticsCheckRes);
                $data['isAnswerCorrect'] = false;
                $componentsCountRes = $this->taskCheckModel->getQueryComponentsCount($correctSolution->t_solution_id);
                $componentsCount = (int)$componentsCountRes->components_count;
                $mistakesCount = count($querySemanticsCheckRes);
                $mark -= $mark * ($mistakesCount / $componentsCount);
                $comment = "Semantics error.";
            }
        } else {
            $data['errorDetail'] = $querySyntaxCheckRes[2];
            $data['isAnswerCorrect'] = false;
            $mark = 0;
            $comment = "Syntax error.";
        }

        /* Check answer end. */

        $dataToInsert = [···
        ];

        if($this->solutionsModel->insert($dataToInsert)) {
            header("Refresh:0");
        } else {
            die('Something went wrong.');
        }
    }

    $this->view('user_pages/labs_tasks/lab_task', $data);
}
```

Рисунок 4.10 – Фрагмент лістингу файлу контролера *Tasks_Student.php*

Логіка перевірки відповіді студента на правильність реалізована у класі моделі *TaskCheck* (рис. 4.11). У методі *checkQuerySyntax($query),* проводиться перевірка на наявність синтаксичних помилок шляхом виклику методу для роботи з навчальною базою даних *tryQuery($query),* що в свою чергу використовує метод *PDO::prepare($sql)* для підготовки запиту до виконання. Така підготовка дозволяє перевірити запит на наявність синтаксичних помилок, не виконуючи його, що є дуже важливим, адже при виконання запитів з відповідей студентів відбувається звернення до єдиного екземпляру навчальної бази даних. Якщо запит синтаксично правильний, то повертається *true*, інакше, повертається повідомлення про помилку, яке передається в контролер, а потім у вигляд.

Метод *checkQuerySemantics($answerQuery,$correct_query_id)* використовується для перевірки семантичної правильності відповіді. У цьому методі, в залежності від типу запиту, відбувається виконання одного з методів перевірки, які передбачають парсинг, форматування та розкладання на компоненти відповіді студента, з подальшим порівнянням кожного компонента наданої відповіді з компонентом правильної відповіді. Результатом, який повертається з методів перевірки за певним типом є масив підказок. Якщо цей масив пустий, то перевірка пройшла успішно, інакше, було виявлено семантичні помилку, тому повертається масив з повідомленнями про компоненти відповіді, які не пройшли перевірку.

```php
<?php
    class TaskCheck {
        private $db;
        private $db_edc;

        public function __construct() {
            $this->db = new Database;
            $this->db_edc = new EducationalDatabase;
        }

        public function checkQuerySyntax($query) {
            return $this->db_edc->tryQuery($query . ";");
        }

        public function checkQuerySemantics($answerQuery, $correct_query_id) {
            $queryType = $this->getQueryType($correct_query_id);
            $queryComponents = $this->getQueryComponents($correct_query_id);
            $hints = [];

            switch(trim($queryType->definition)) {
                case "SELECT SIMPLE":
                    $this->simpleSelectCheck($answerQuery, $queryComponents, $hints);
                    break;
                case "SELECT JOIN":
                    $this->joinSelectCheck($answerQuery, $queryComponents, $hints);
                    break;
                case "CREATE":
                    $this->createCheck($answerQuery, $queryComponents, $hints);
                    break;
                case "PLAIN TEXT":
                    $this->plainTextCheck($answerQuery, $queryComponents, $hints);
                    break;
            }

            if(empty($hints)) {
                return true;
            } else {
                return $hints;
            }
        }

        private function getQueryType($query_id) {···
        }
```

Рисунок 4.11 – Фрагмент лістингу файлу моделі *TaskCheck.php*

Для перегляду та підтвердження/відхилення відповідей студентів викладачем, використовуються методи *new_solutions()* та *new_solution($solution_id)* контролера *Solutions* (рис. 4.12). Перший з них отримує з моделі усі нові (непідтверджені) відповіді студентів та передає їх у вигляд для заповнення відповідної таблиці. Другий – отримує певну з моделі окрему відповідь, яку необхідно відредагувати та завантажує вигляд з формою, шо містить відомості про відповідь студента, з можливістю зміни оцінки та/або коментаря до відповіді. Крім цього, другий метод передбачає валідацію введеної викладачем оцінки.

```php
<?php
    require(APPROOT . '/libraries/Excel/PHPExcel.php');

    trait Solutions {
        private $solutionsModel;

        public function new_solutions() {
            $solutions = $this->solutionsModel->getNewStudentsSolutions();
            $groups = $this->groupModel->getAllGroups();
            $data = [
                'solutions' => $solutions,
                'groups' => $groups
            ];

            $this->view('admin_pages/solutions/new_solutions', $data);
        }

        public function new_solution($solution_id) {
            $solution = $this->solutionsModel->getNewStudentsSolutionById($solution_id);
            $data = [···
            ];

            if($_SERVER['REQUEST_METHOD'] == 'POST') {
                $_POST = filter_input_array(INPUT_POST, FILTER_SANITIZE_STRING);

                $data = [···
                ];

                $markRule = "/^[0-9.]{1,5}$/";

                if($data['mark'] === '') {
                    $data['markError'] = 'Enter the mark value, please.';
                } elseif(!preg_match($markRule, $data['mark'])) {
                    $data['markError'] = 'Mark must be a positive integer or floating point number.';
                } elseif(floatval($data['mark']) > $data['solution']->difficulty) {
                    $data['markError'] = 'Mark can not be bigger then maximum value.';
                }

                if(!$data['markError']) {
                    if($this->solutionsModel->saveConfirm(array_slice($data, 0, 3))) {
                        header('location:' . URLROOT . '/admin_pages/solutions/new_solutions');
                    } else {
                        die("Something went wrong...");
                    }
                }
            }

            $this->view('admin_pages/solutions/new_solution', $data);
        }
```

Рисунок 4.12 – Фрагмент лістингу файлу контролера *Solutions.php*

Клас моделі *Solution* (рис. 4.13) використовується для проведення ряду операцій з відповідями студентів (новими/підтвердженими) та правильними відповідями, встановленими викладачем. Наприклад, для отримання нових відповідей використовується метод *getNewStudentSolutions()*, в якому формується та виконується запит до декількох таблиць, за для формування більш повної інформації про надіслані відповіді, а метод *getNewStudentSolutionById($id)*, виконує аналогічний запит, проте відбувається пошук лише однієї відповіді – що має вказаний ідентифікатор.

```php
<?php
    class Solution {
        private $db;

        public function __construct() {
            $this->db = new Database;
        }

        public function getNewStudentsSolutions() {
            $this->db->setQuery('SELECT ss.s_solution_id AS s_solution_id, ss.answer AS answer, ss.mark AS mark, ss.comment AS comment, t.number AS task_number,
                                        t.difficulty AS difficulty, lw.title_en AS lab_title, u.name AS name, u.surname AS surname, g.number AS group_number
                                FROM `student_solutions` AS ss
                                LEFT JOIN `tasks` AS t ON ss.task_id = t.task_id
                                LEFT JOIN `laboratory_works` AS lw ON ss.lab_id = lw.lab_id
                                LEFT JOIN `users` AS u ON ss.user_id = u.user_id
                                LEFT JOIN `groups` AS g ON u.group_id = g.group_id
                                WHERE ss.confirmed = :confirmed
                                ORDER BY answer_date ASC');

            $this->db->bindQueryParameter(':confirmed', 0);

            $resultRows = $this->db->getArrayResult();
            return $resultRows;
        }

        public function getNewStudentsSolutionById($id) {
            $this->db->setQuery('SELECT ss.s_solution_id AS s_solution_id, ss.answer AS answer, ss.answer_date AS answer_date, ss.mark AS mark,
                                        ss.comment AS comment, t.number AS task_number, t.exercise_en AS exercise, t.difficulty AS difficulty,
                                        lw.title_en AS lab_title, u.name AS name, u.surname AS surname, g.number AS group_number
                                FROM `student_solutions` AS ss
                                LEFT JOIN `tasks` AS t ON ss.task_id = t.task_id
                                LEFT JOIN `laboratory_works` AS lw ON ss.lab_id = lw.lab_id
                                LEFT JOIN `users` AS u ON ss.user_id = u.user_id
                                LEFT JOIN `groups` AS g ON u.group_id = g.group_id
                                WHERE ss.s_solution_id = :solution_id AND ss.confirmed = :confirmed');

            $this->db->bindQueryParameter(':solution_id', $id);
            $this->db->bindQueryParameter(':confirmed', 0);

            $resultRow = $this->db->getSingleResult();
            return $resultRow;
        }

        public function getCheckedStudentsSolutions() {···
        }
```

Рисунок 4.13 – Фрагмент лістингу файлу моделі *Solution.php*

Для більш швидкого пошуку необхідних записів, до таблиці з новими відповідями, як і до ряду інших, було додано фільтри за певними атрибутами, що реалізовані у файлі *solutions.table.js* (рис. 4.14). Для таблиці з новими відповідями цими атрибутами є ім’я та прізвище виконавця, його група, лабораторна робота і номер завдання. У файлах вигляду ці фільтри реалізовані як поля для пошуку та розкривний список (для групи). Фільтрація за атрибутами для яких встановлено поля для пошуку відбувається одразу після введення першого символу. Одночасне встановлення значень фільтрів для декількох стовпців проводить фільтрацію одночасно за всіма значеннями. Для скасування фільтрації за певним атрибутом достатньо очистити відповідне поле пошуку.

```
"use strict";

const solutionsTableFilters = {
    nameSurnameFilterVal: "",
    groupFilterVal: "",
    labworkFilterVal: "",
    taskFilterVal: ""
};

$(document).ready(function() {
    $("#nameSurnameFilter").on("keyup", function() {
        solutionsTableFilters.nameSurnameFilterVal = $(this).val().toLowerCase();
    });
    $("#groupFilter").on("change", function() {
        solutionsTableFilters.groupFilterVal = $(this).val();
    });
    $("#labFilter").on("keyup", function() {
        solutionsTableFilters.labworkFilterVal = $(this).val().toLowerCase();
    });
    $("#taskFilter").on("keyup", function() {
        solutionsTableFilters.taskFilterVal = $(this).val().toLowerCase();;
    });

    $("#nameSurnameFilter, #labFilter, #taskFilter").on("keyup", tableFilter);
    $("#groupFilter").on("change", tableFilter);

    $("#selectAll").on("click", checkAll);

    $("td input:checkbox").on("change", changeRowBackgroundOnCheck);

    $("#solutionProcessForm").on("submit", setSolutionsIdsOnProcessing);
});

function tableFilter() {
    $("#solutionsTBody tr").filter(function() {
        let isNameSurnameMatch = $(this).children('.s-namesurname').text().toLowerCase().indexOf(solutionsTableFilters.nameSurnameFilterVal) > -1;
        let isGroupMatch = ($(this).children('.s-group').text().trim() === solutionsTableFilters.groupFilterVal.trim()) || !solutionsTableFilters.groupFilterVal.trim();
        let isLabworkMatch = $(this).children('.s-lab').text().toLowerCase().indexOf(solutionsTableFilters.labworkFilterVal) > -1;
        let isTaskMatch = $(this).children('.s-task').text().toLowerCase().indexOf(solutionsTableFilters.taskFilterVal) > -1;

        $(this).toggle(isNameSurnameMatch && isGroupMatch && isLabworkMatch && isTaskMatch);
    });
}
```

Рисунок 4.14 – Фрагмент лістингу файлу *solutions.table.js*

4.3 Використання інформаційної системи

Перша сторінка, на яку потрапляє користувач при завантаженні розробленого web-застосунку – це сторінка авторизації, що містить форму (рис. 4.15), на якій йому необхідно ввести, визначені при реєстрації логін та пароль, щоб отримати доступ до системи. Крім, цього, під час авторизації визначається роль користувача у системі: викладач або студент.

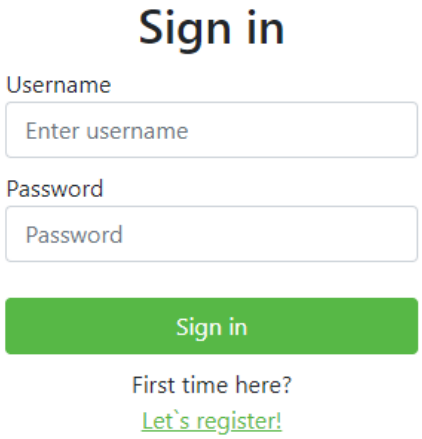


Рисунок 4.15 – Форма авторизації

У разі введення неправильної інформації, наприклад, якщо користувача із зазначеним логіном не існує, або якщо пароль користувача з таким логіном введено невірно, то при натисканні на кнопку «Sign in» користувач отримує відповідне повідомлення про помилку при авторизації (рис. 4.16).

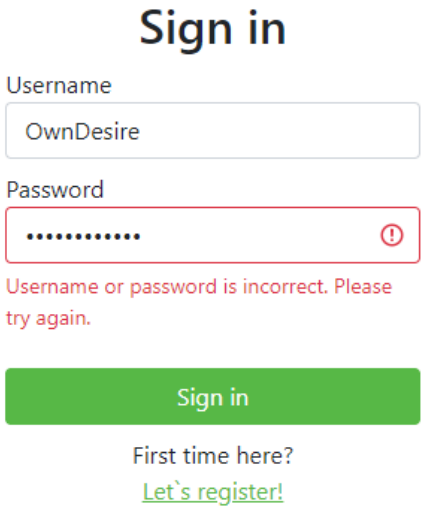


Рисунок 4.16 – Перевірка даних під час авторизації

Якщо користувач-студент вперше користується системою, то йому необхідно зареєструватися натиснувши на підпис «Let`s register!», що розташований внизу форми авторизації. Після цього, він потрапляє на сторінку реєстрації, що містить форму (рис. 4.17), на якій йому необхідно ввести своє ім’я, прізвище, обрати групу, визначити логін та пароль, та підтвердити пароль, ввівши його повторно; або, якщо в

користувач вже зареєстрований – повернутися на сторінку авторизації натиснувши на напис «Let`s sign in!».

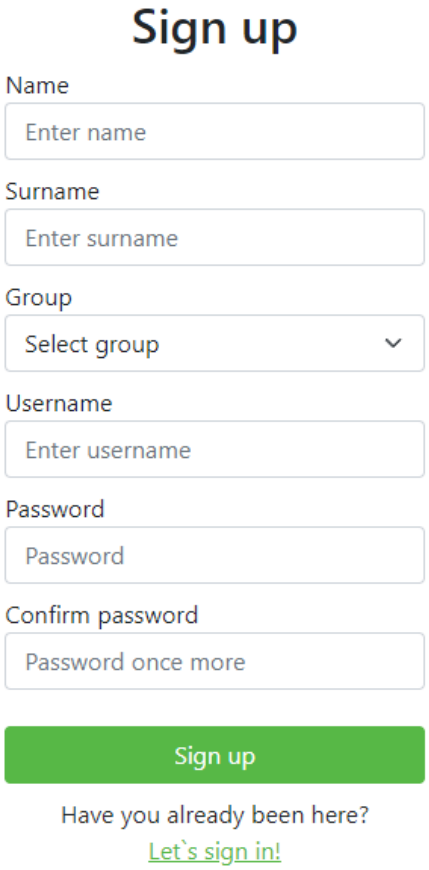


Рисунок 4.17 – Форма реєстрації

Для форми реєстрації також передбачена валідація введеної інформації, наприклад, ім’я та прізвище можуть складатися лише з літер латинського алфавіту та кирилиці, а логін та пароль можуть містити лише літери латинського алфавіту, та їх довжина не повинна перевищувати визначеного у системі значення. У разі натискання на кнопку «Sign up», недотримавшись визначених обмежень, відображається відповідне повідомлення про помилку (рис. 4.18).

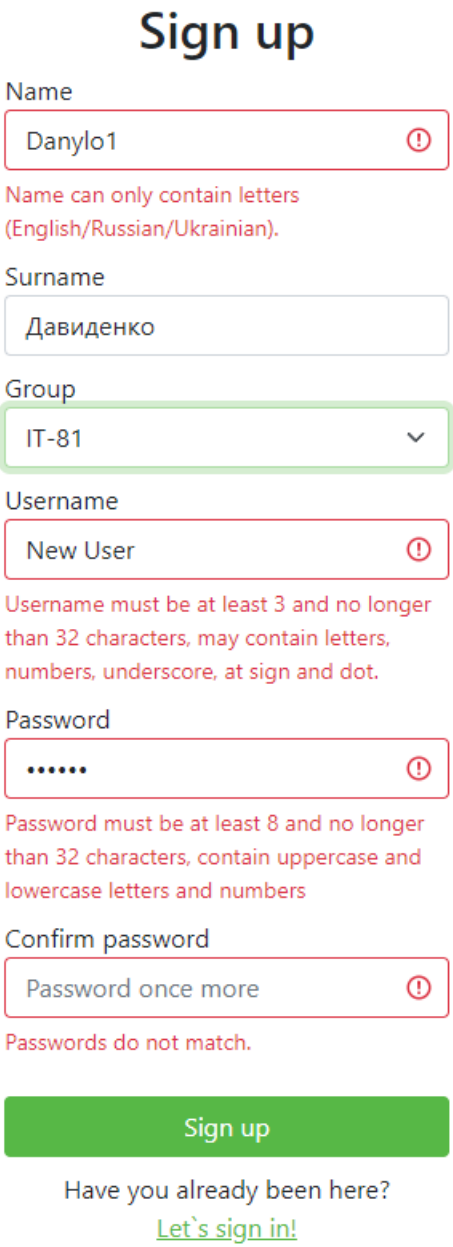


Рисунок 4.18 – Перевірка даних під час реєстрації

Користувач-студент, який успішно пройшов реєстрацію отримує доступ до системи не одразу, йому необхідно очікувати на підтвердження від викладача (адміністратора), а при спробі авторизуватися без, наданого викладачем, доступу, він отримує відповідне повідомлення (рис. 4.19).

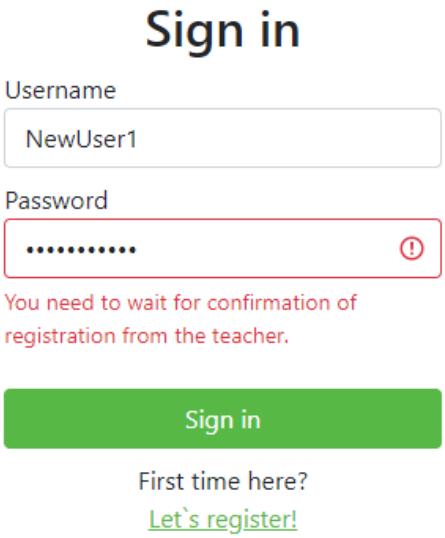


Рисунок 4.19 – Перевірка наявності прав на авторизацію

Після отримання доступу до системи, користувач може успішно авторизуватися та потрапити на головну сторінку, на якій відображається таблиця з інформацією про лабораторні роботи: назва роботи, дати початку та завершення, а також його поточною оцінкою за кожну з цих робіт (рис. 4.20). Крім цього, у верхній частині цієї сторінці (як і на інших) відображається навігаційне меню, за допомогою якого користувач може перейти на поточну сторінку з лабораторними роботами, на сторінку зі структурою навчальної бази даних (до якої робляться запити при виконанні завдань), перемкнути мову відображення інтерфейсу (Англійська або Українська) та вийти з системи.

Для того, щоб перейти до певного завдання, користувачу необхідно обрати лабораторну роботу та натиснути на відповідний рядок таблиці з лабораторними роботами. Таким чином, він потрапляє на сторінку вибору завдання обраної лабораторної роботи (рис. 4.21). На цій сторінці відображається інформація про кожне завдання лабораторної роботи: номер, зміст, складність, скільки залишилось спроб та результативність кращої спроби.

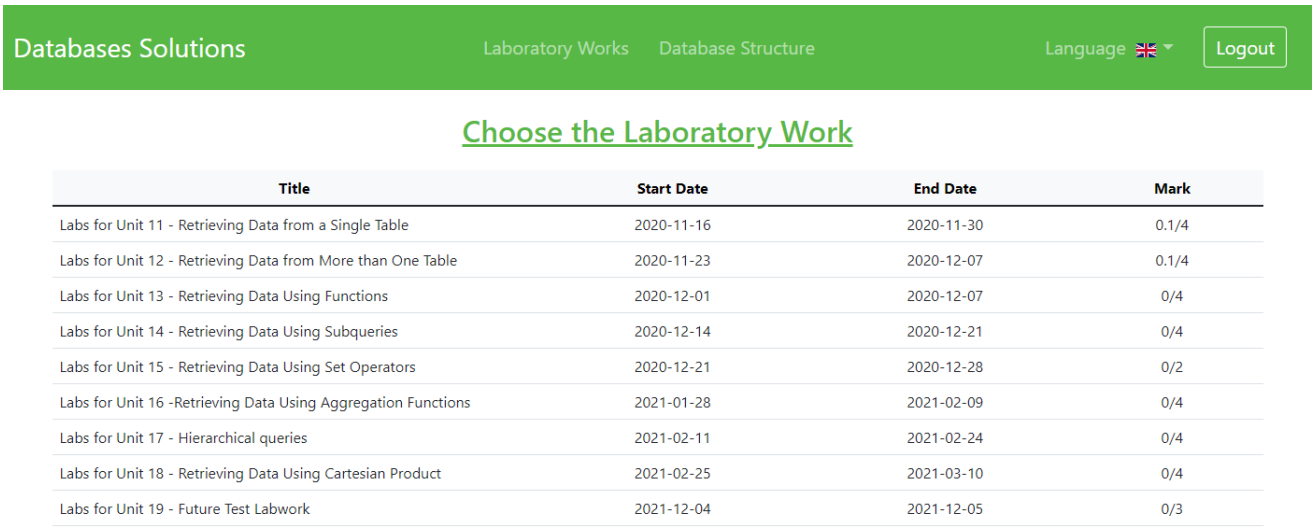


Рисунок 4.20 – Сторінка з таблицею лабораторних робіт для користувача-студента

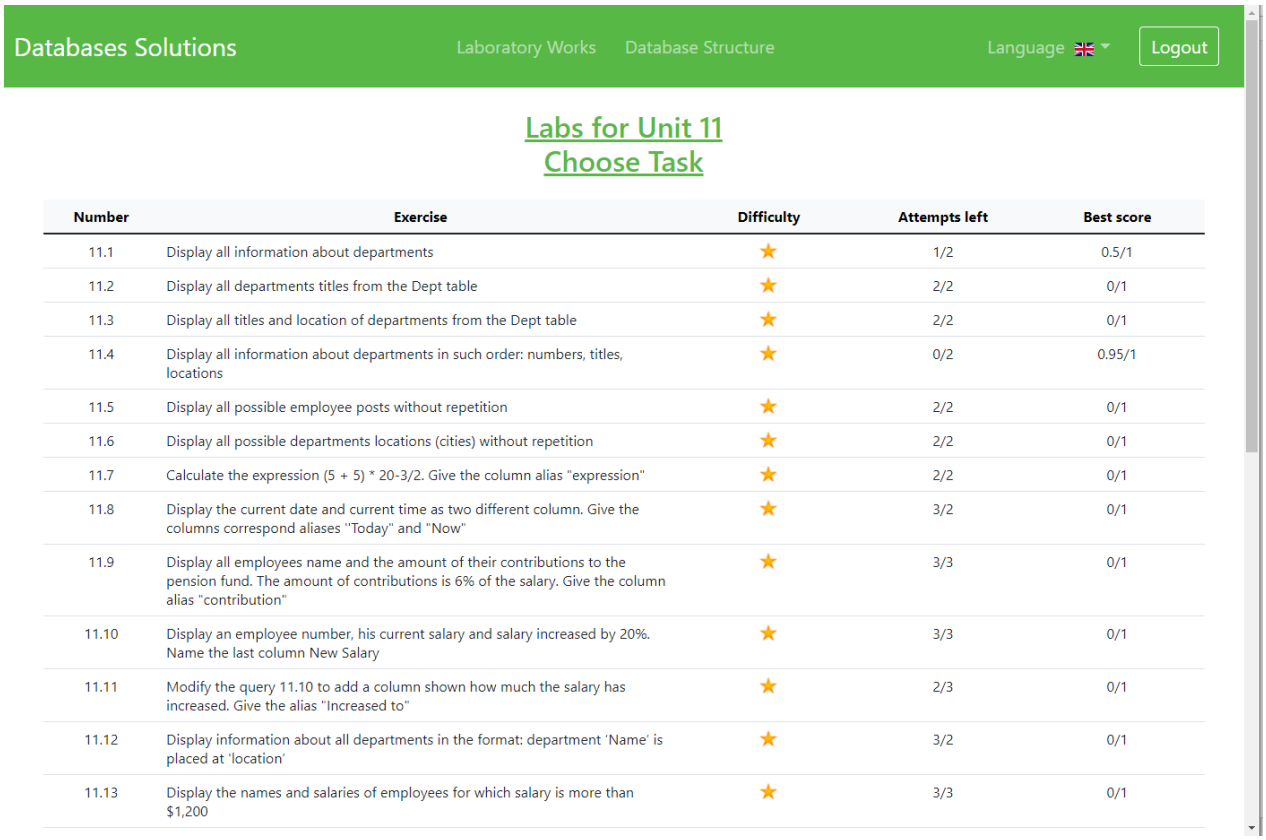


Рисунок 4.21 – Сторінка з таблицею завдань до обраної лабораторної роботи для користувача-студента

Натиснувши на певне завдання з таблиці користувач потрапляє на сторінку для виконання завдання (рис. 4.22). На цій сторінці відображається зміст завдання, його складність, скільки спроб залишилось у користувача на виконання, поле для введення

відповіді, кнопки для швидкого введення шаблонів найбільш уживаних команд, кнопка для очищення поля відповіді, кнопка для підтвердження відповіді, а також таблиця зі списком спроб користувача за цим завданням.

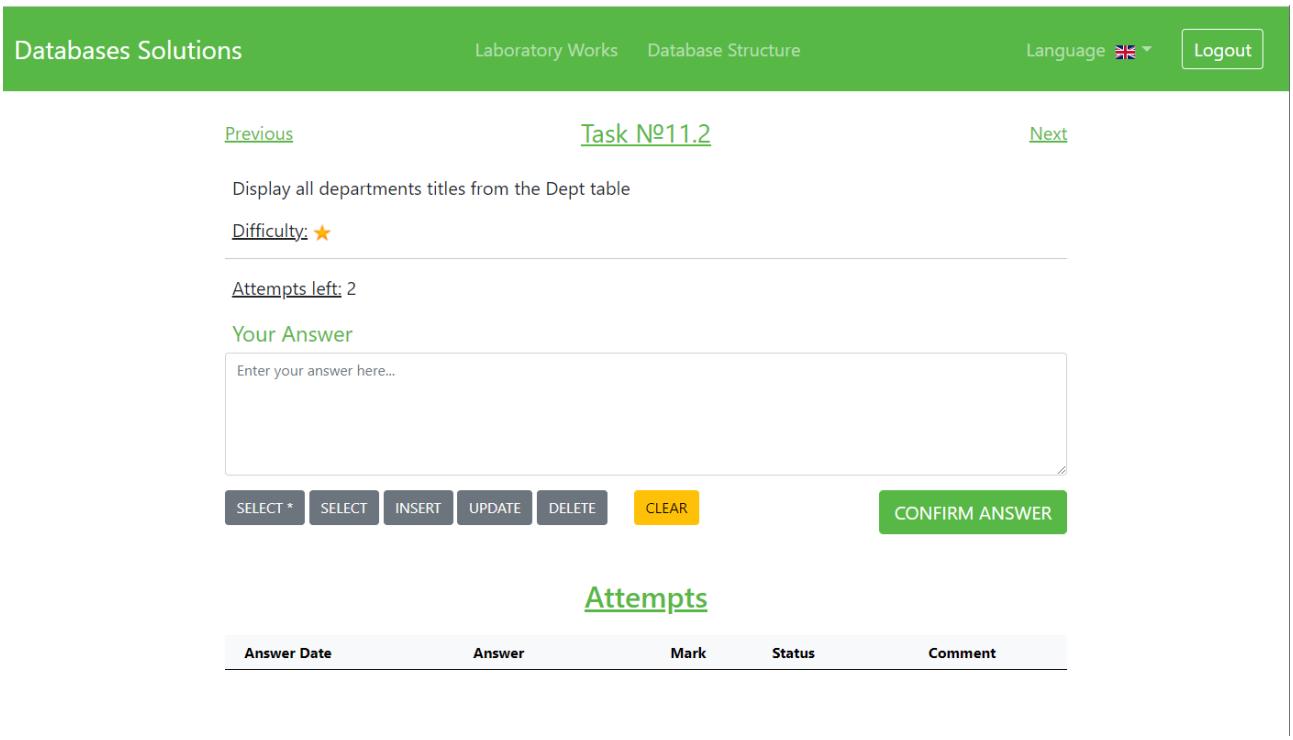


Рисунок 4.22 – Початковий вигляд сторінки виконання обраного завдання

Прочитавши, проаналізувавши та визначившись з відповіддю до завдання, користувач має ввести цю відповідь у відповідне поле. Якщо відповідь має синтаксичні помилки (згідно синтаксису SQL запитів для системи керування базами даних MySQL та обмежень навчальної бази даних), то відображається відповідне повідомлення, а в таблицю спроб заноситься запис із нульовим значенням оцінки та коментарем про те, що під час перевірки відповіді (SQL запиту) було знайдено синтаксичні помилки (рис. 4.23). У випадку, якщо відповідь не містить синтаксичних помилок, то відбувається перевірка на наявність семантичних помилок, і у разі виявлення таких помилок, відображається відповідне повідомлення, а до таблиці зі спробами заноситься знижена або мінімальна оцінка та відповідний коментар. Крім цього, якщо користувач вичерпав свої спроби , то кнопка «CONFIRM ANSWER» стає неактивною, і надсилання відповідей за цим завданням стає неможливим (рис 4.24).

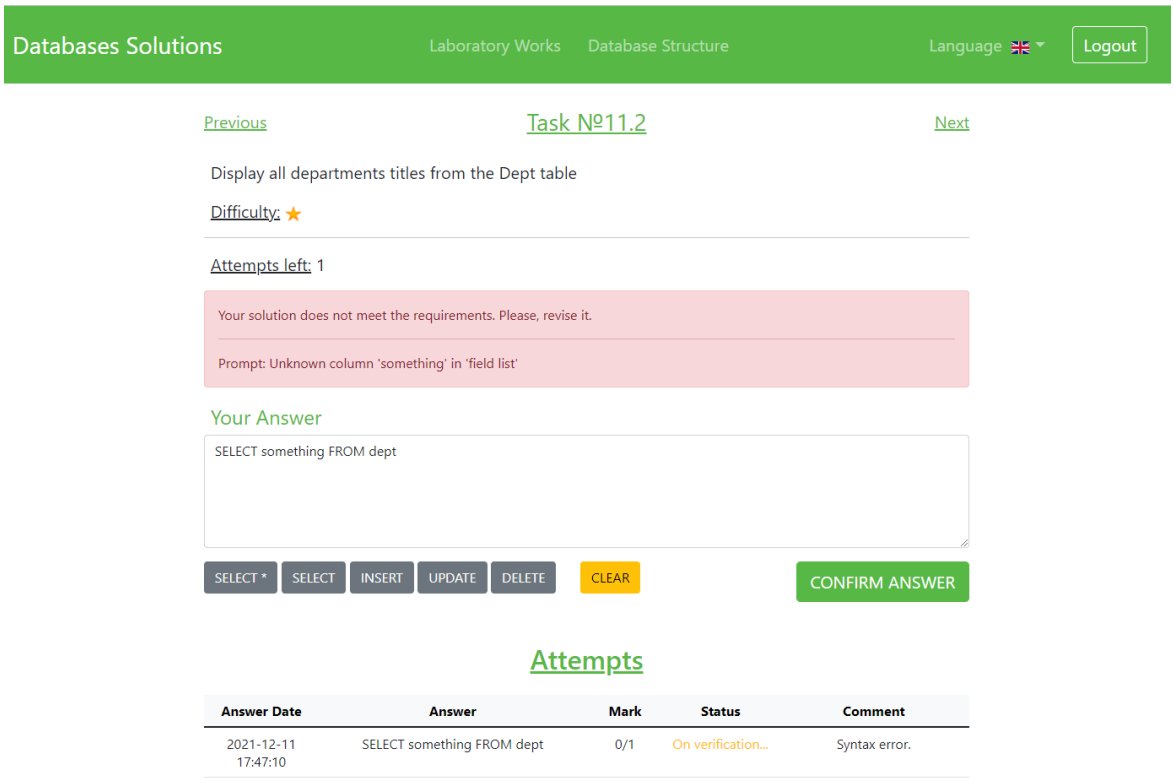


Рисунок 4.23 – Відображення повідомлення у разі знаходження синтаксичної помилки у відповіді

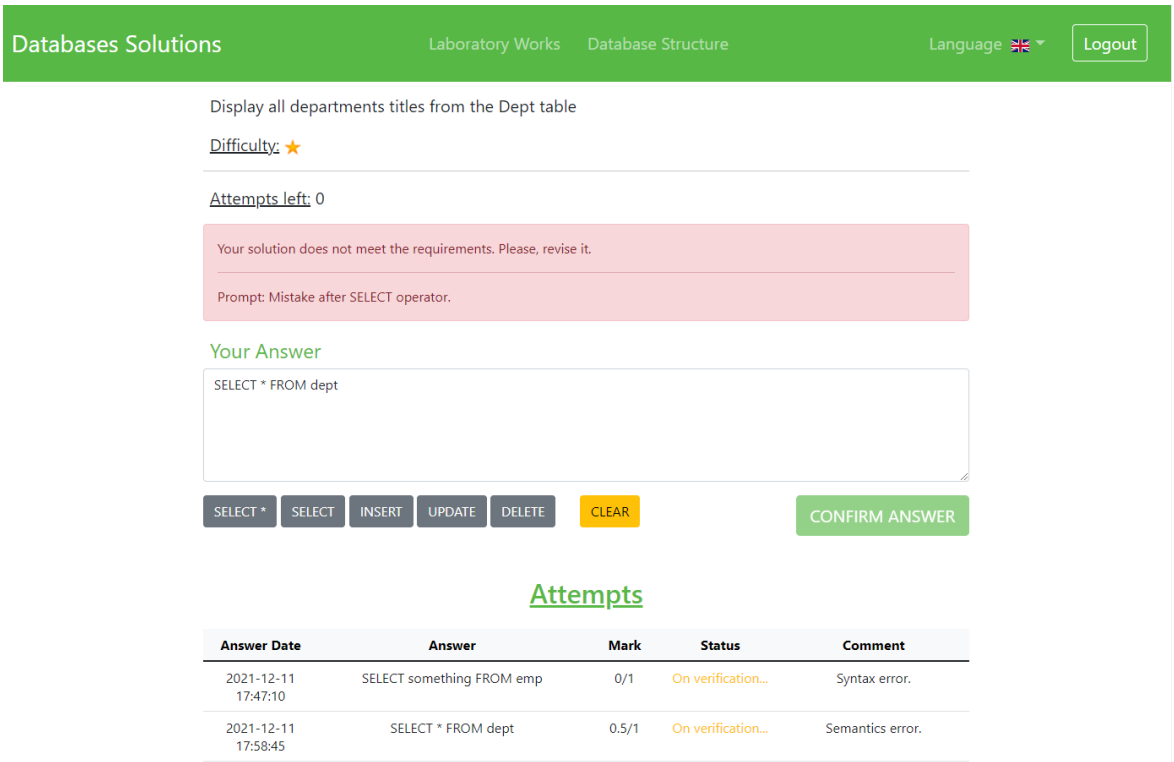


Рисунок 4.24 – Відображення повідомлення у разі знаходження семантичної помилки у відповіді

При введенні правильної відповіді, користувач отримує відповідне повідомлення, а у таблицю спроб заноситься максимальний бал та відповідний коментар (рис 4.25).

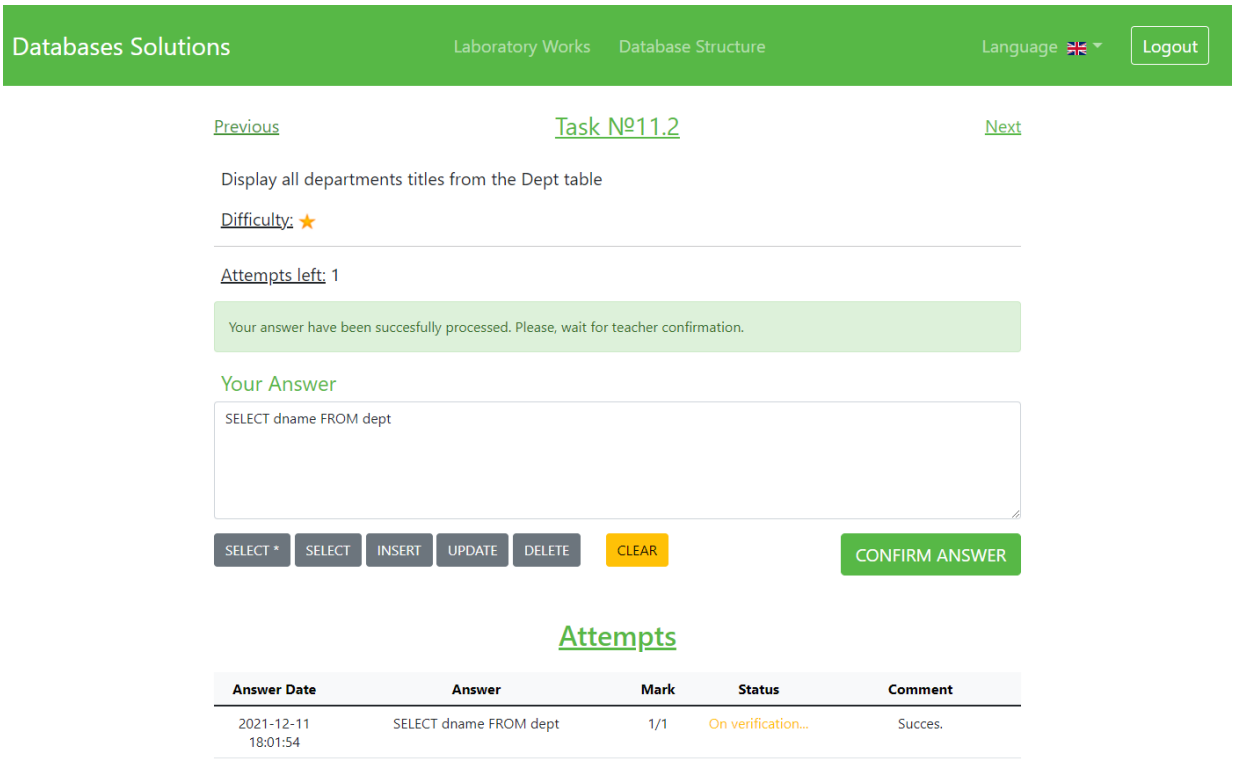


Рисунок 4.25 – Відображення повідомлення про правильність наданої відповіді

Тепер, розглянемо можливості користувача-викладача (адміністратора). Після успішної авторизації викладач потрапляє на сторінку зі списком лабораторних робіт, на якій відображається загальна інформація по кожній з робіт: назва, дати початку та завершення виконання, максимальний бал, а також кнопка для створення нової лабораторної роботи (рис. 4.26). У верхній частині розташовано навігаційне меню, яке дозволяє викладачу перейти на поточну сторінку, сторінки керування інформацією про користувачів, сторінки керування надісланими відповідями студентів, сторінку зі структурою навчальної бази даних, змінити мову відображення інтерфейсу (Англійська або Українська) та вийти з системи.

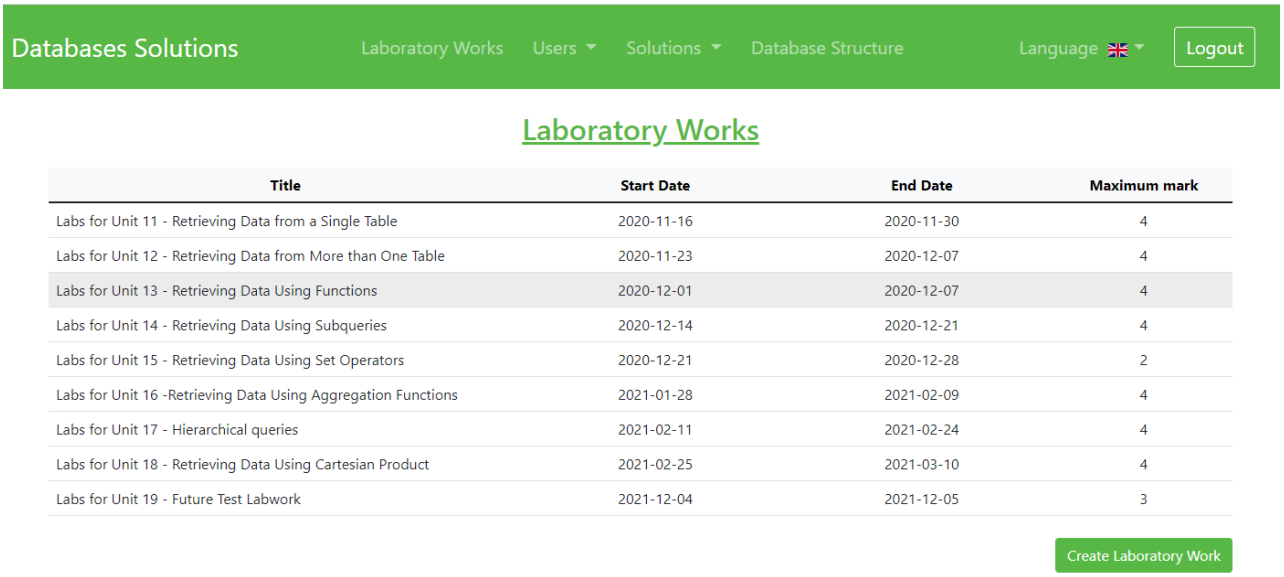


Рисунок 4.26 – Сторінка з таблицею лабораторних робіт для користувача-викладача

У разі натискання кнопки «Create Laboratory Work» відкривається сторінка з формою для створення нової лабораторної роботи (рис 4.27). Після натискання на кнопку «Create», інформація введена у поля даної форми проходить валідацію, і у випадку, якщо було введено недопустимі дані, або необхідні дані відсутні, то відображаються відповідні підказки-помилки, а лабораторна роботи не може бути створена. Якщо введені дані пройшли валідацію – робота додається успішно.

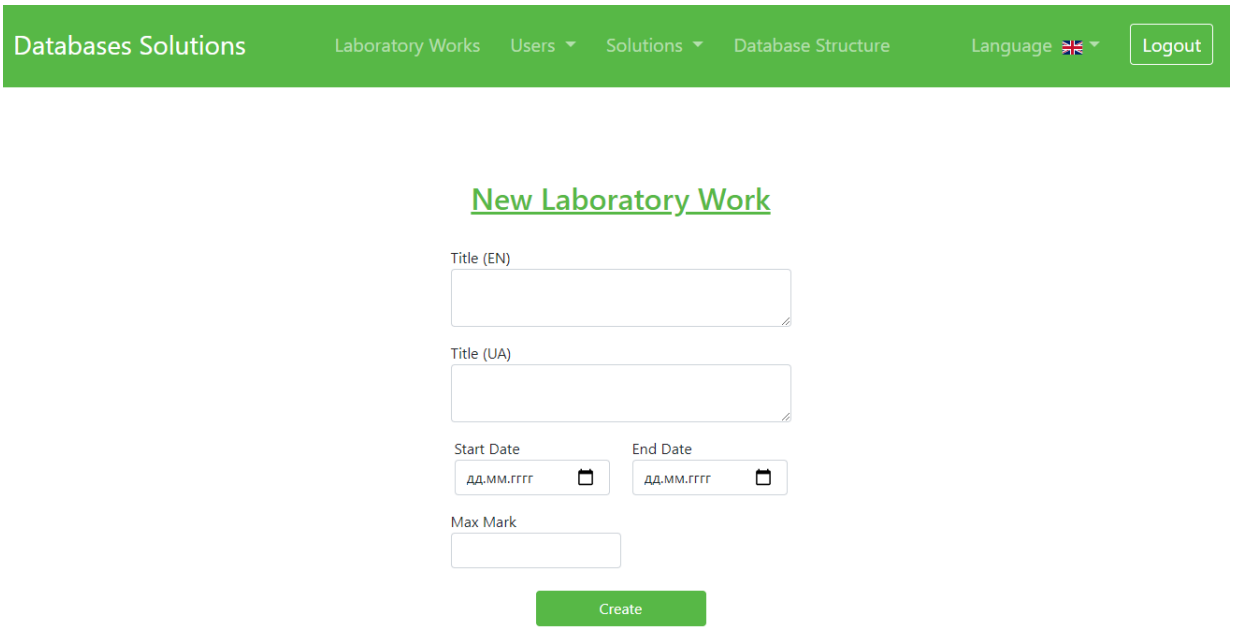


Рисунок 4.27 – Сторінка створення нової лабораторної роботи

При натисканні на певний рядок таблиці з лабораторними роботами відкривається сторінка з формою для редагування або видалення даних цієї роботи та завданнями, які вона включає (рис. 4.28). Форма для редагування даних лабораторної роботи також передбачає валідацію введеної інформації, тому при спробі збереження з недопустимими даними відобразяться підказки-помилки, а зміни не будуть збережені.

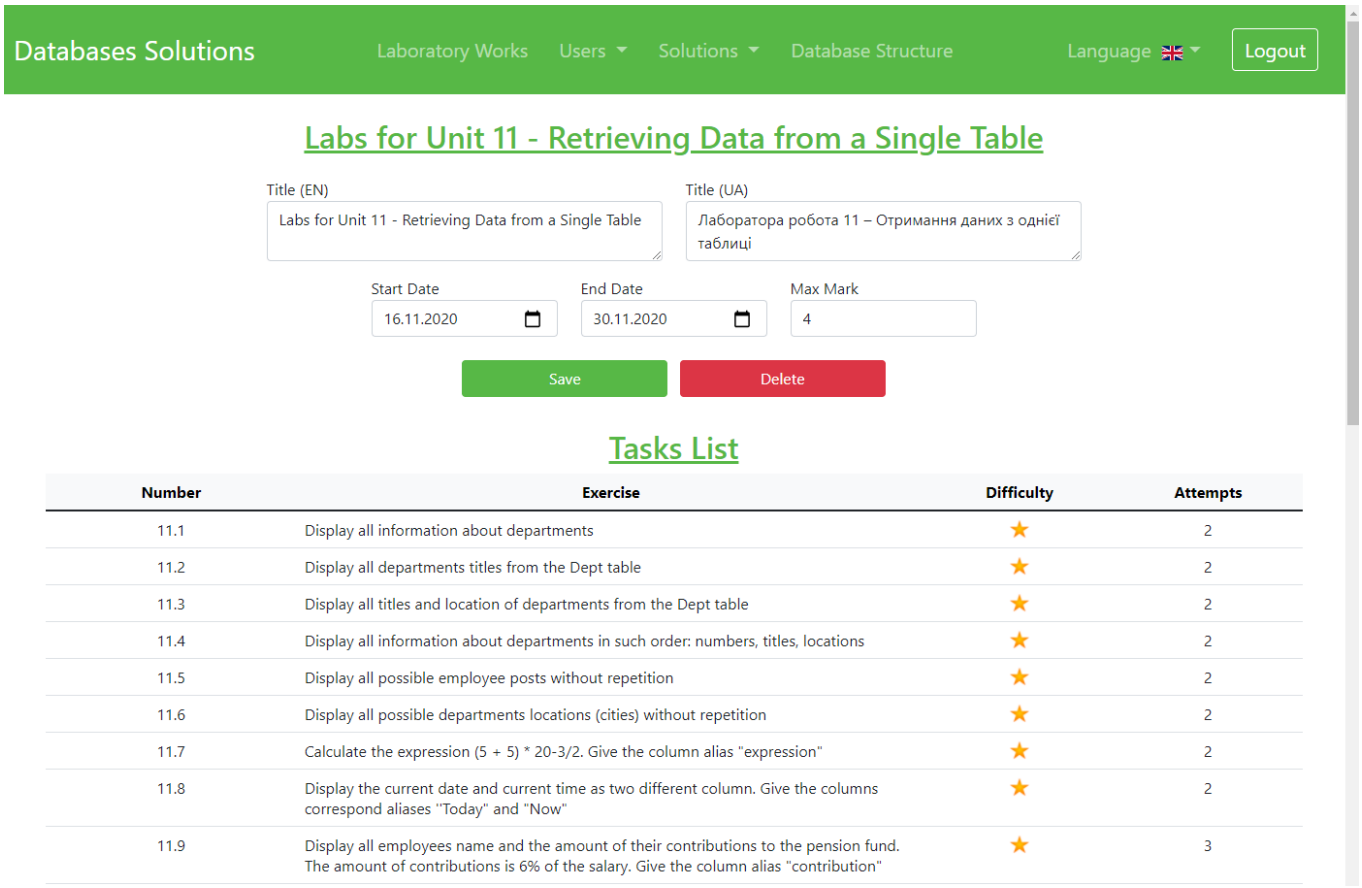


Рисунок 4.28 – Сторінка для редагування лабораторної роботи та перегляду її завдань

Внизу цієї сторінки розташована кнопка додавання завдання до цієї лабораторної роботи, натискання якої передбачає відкриття сторінки з формою для створення нового завдання. При натисканні на певний рядок таблиці із завданнями відображається сторінка з формою редагування обраного завдання (рис. 4.29). Обидві форми: для створення та для редагування/видалення завдання, передбачать валідацію введених даних.

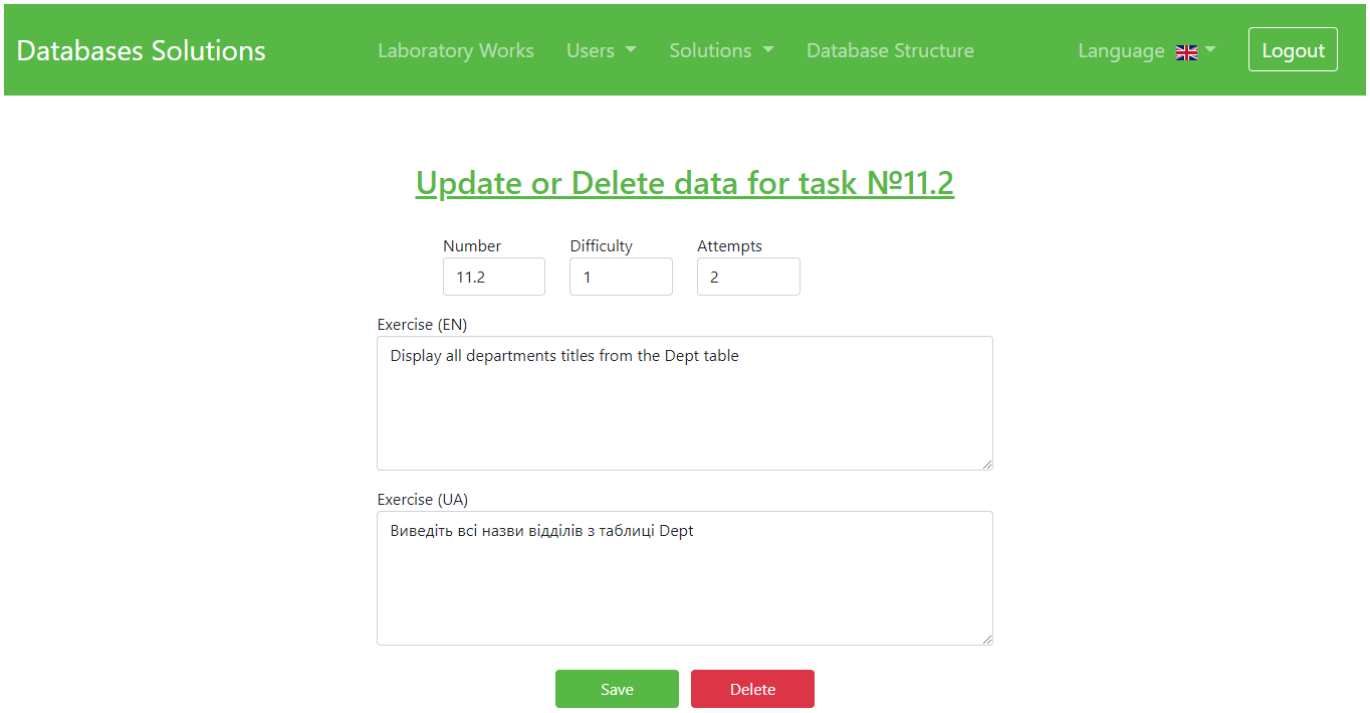


Рисунок 4.29 – Сторінка для редагування інформації про завдання

При переході, за допомогою навігаційного меню, на сторінку, де відображаються нові користувачі системи (які зареєструвалися, але не отримали підтвердження), відображається відповідна таблиця та дві кнопки: підтвердження та відхилення (рис. 4.30). Для того, щоб підтвердити або відхили реєстрацію користувача-студента, викладач має обрати користувача/користувачів зі списку шляхом встановлення перемикача в активне положення та натиснути кнопку «Accept selected» – для підтвердження реєстрації або «Decline selected» – для відхилення (видалення) користувача.

Окрім сторінки з новими користувачами, викладач може перейти на сторінку з наявними користувачами системи. На ній відображається дві таблиці: викладачі/адміністратори та студенти (рис. 4.31). Таблиці на сторінках з новими та наявними користувачами системи дозволяють проводити фільтрацію для більш швидкого пошуку необхідних користувачів.

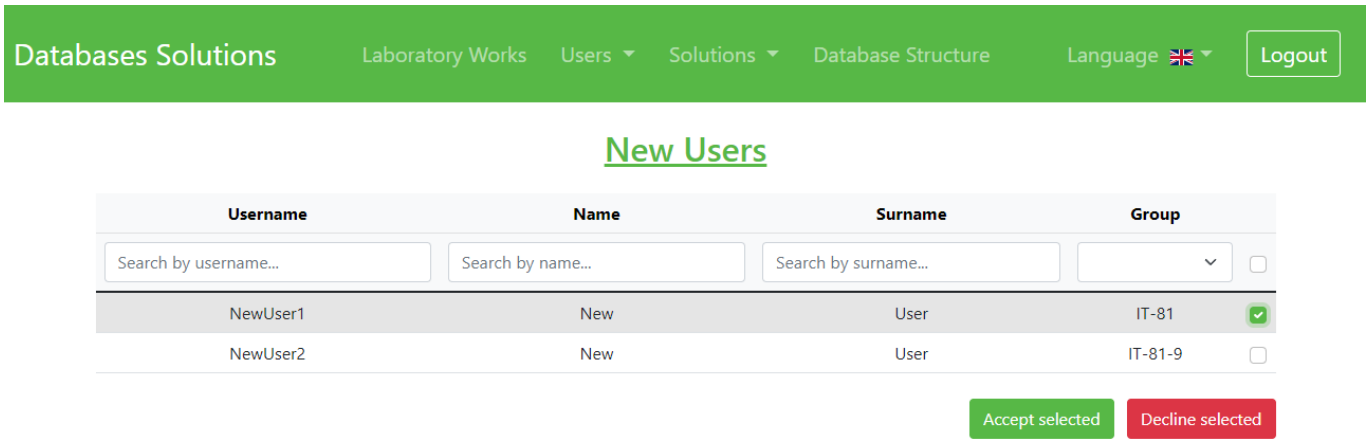


Рисунок 4.30 – Сторінка підтвердження/відхилення нових користувачів

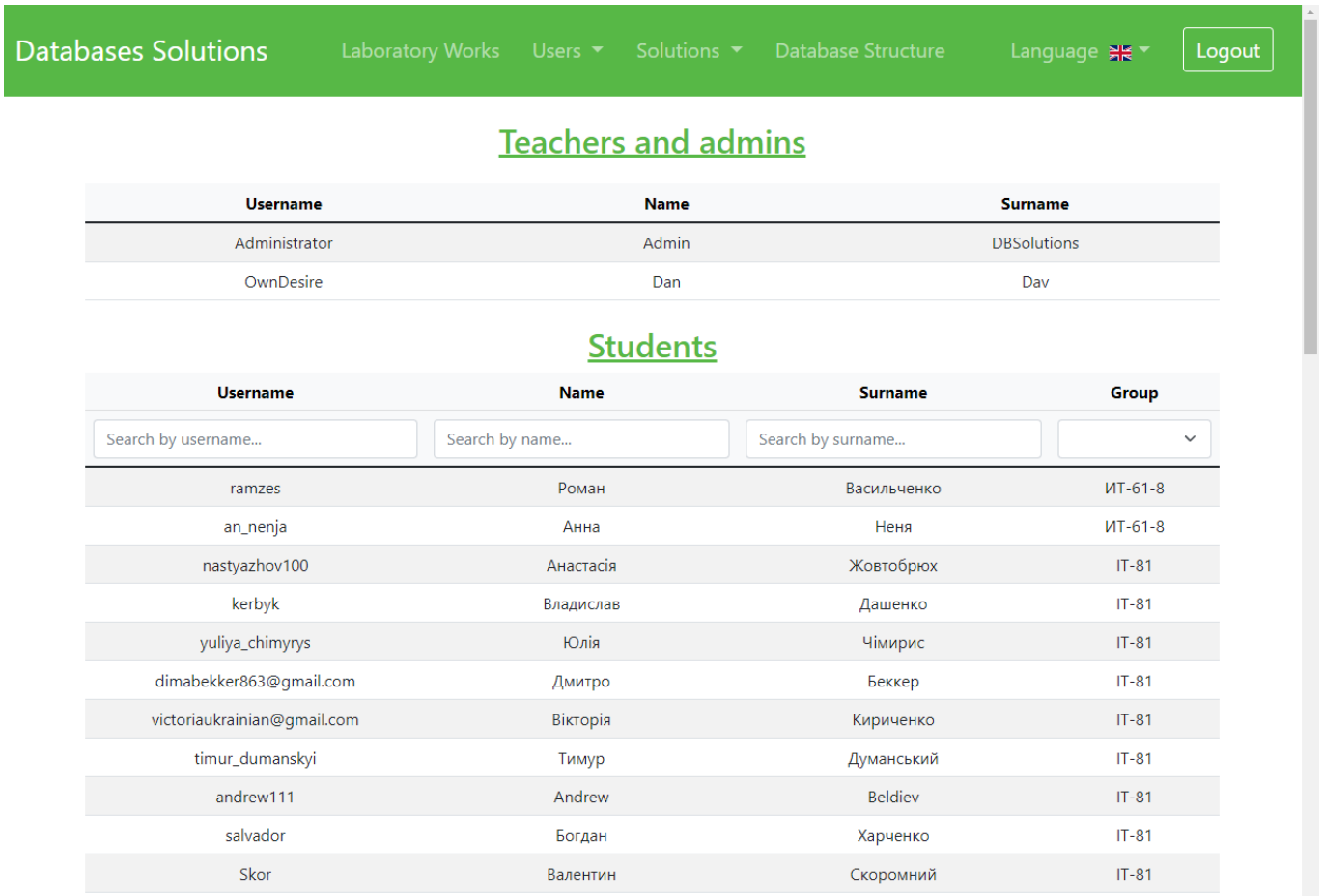


Рисунок 4.31 – Сторінка з підтвердженими користувачами системи

Натиснувши на певний рядок будь-якої з двох таблиць з підтвердженими користувачами системи, викладач переходить на сторінку з формою для редагування/видалення інформації певного користувача (рис. 4.32). Окрім звичайних

функцій: збереження змін та видалення, на цій формі наявна опція зробити користувача адміністратором, що може стати у нагоді, наприклад, якщо буде необхідність додати нового викладача, що реєструватиметься як звичайний користувач. Форма передбачає валідацію (аналогічно обмеженням відповідних полів при реєстрації) при спробі збереження змін.

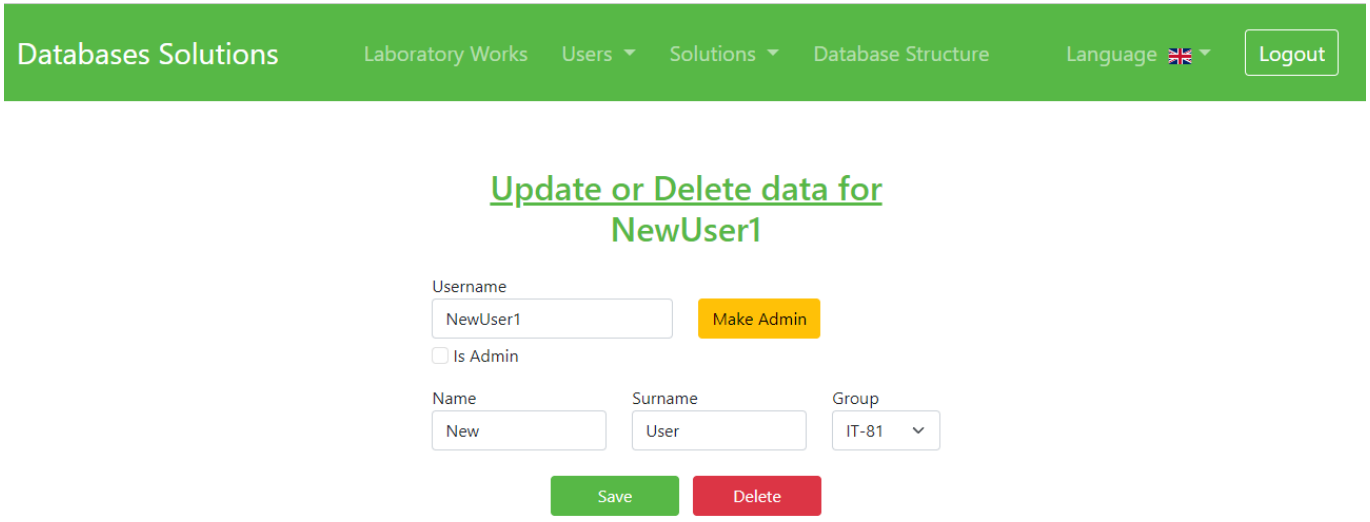


Рисунок 4.32 – Сторінка з підтвердженими користувачами системи

Відповіді надіслані студентами відображаються у викладача на двох сторінка. На одній з них відображаються нові відповіді, ні іншій – підтверджені відповіді. На сторінці з новими відповідями, окрім таблиці з відповідями студентів, наявні дві кнопки, за допомогою яких можна підтвердити або відхилити одразу декілька відповідей (рис. 4.33). При необхідності можна переглянути більш детальну інформацію не підтвердженої відповіді студента, натиснувши на відповідний рядок таблиці. Після цього відкривається сторінка редагування оцінки та коментаря до відповіді студента (рис. 4.34).

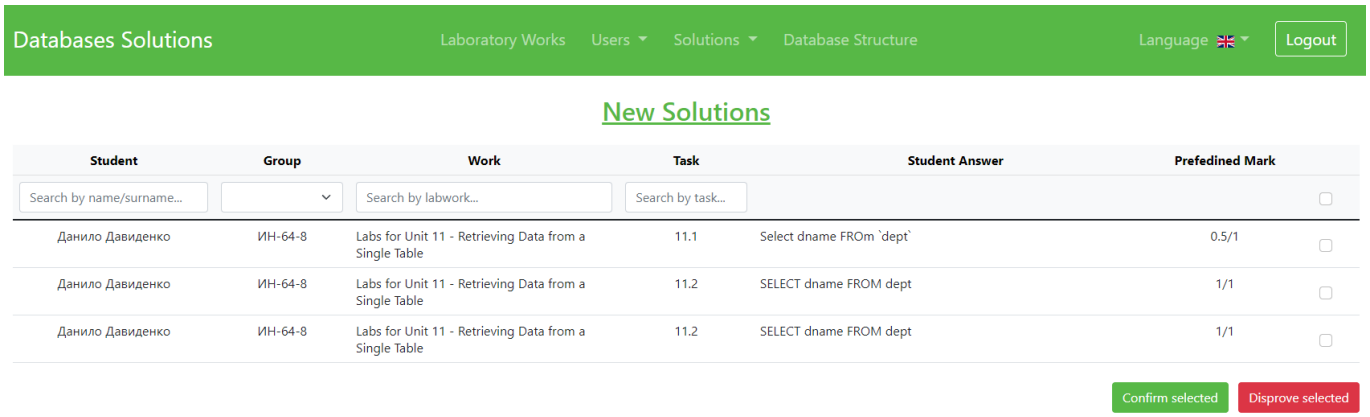


Рисунок 4.33 – Сторінка з новими відповідями студентів

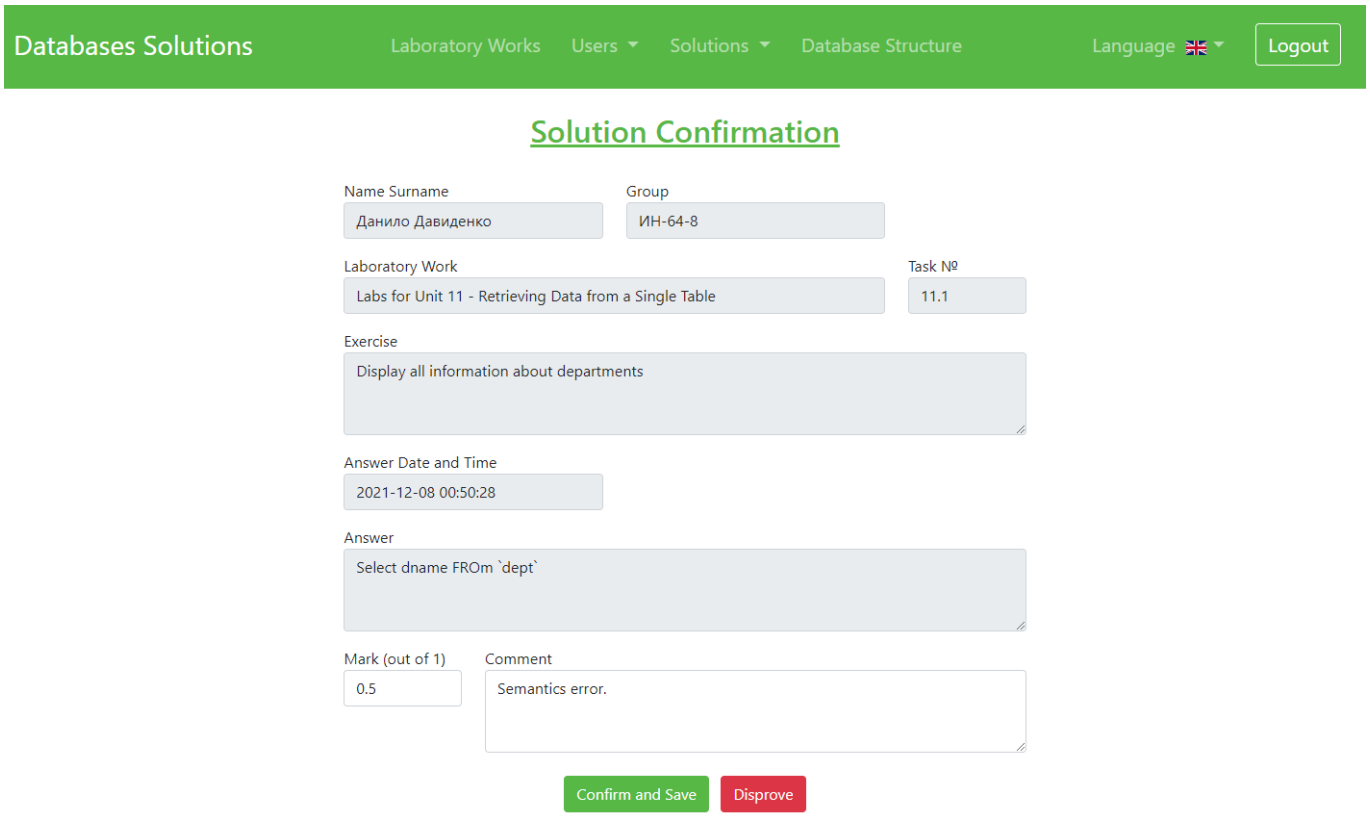


Рисунок 4.34 – Сторінка редагування оцінки та коментаря до нової відповіді студента

На сторінці з перевіреними відповідями студентів знаходиться таблиця з відповідями та кнопка для формування звіту за певний рік (рис. 4.35), що передбачає експорт оцінок студентів за лабораторні роботи у документ MS Excel. Таблиці з новими та перевірені відповідями студентів мають фільтри для більш швидкого пошуку необхідних відповідей.

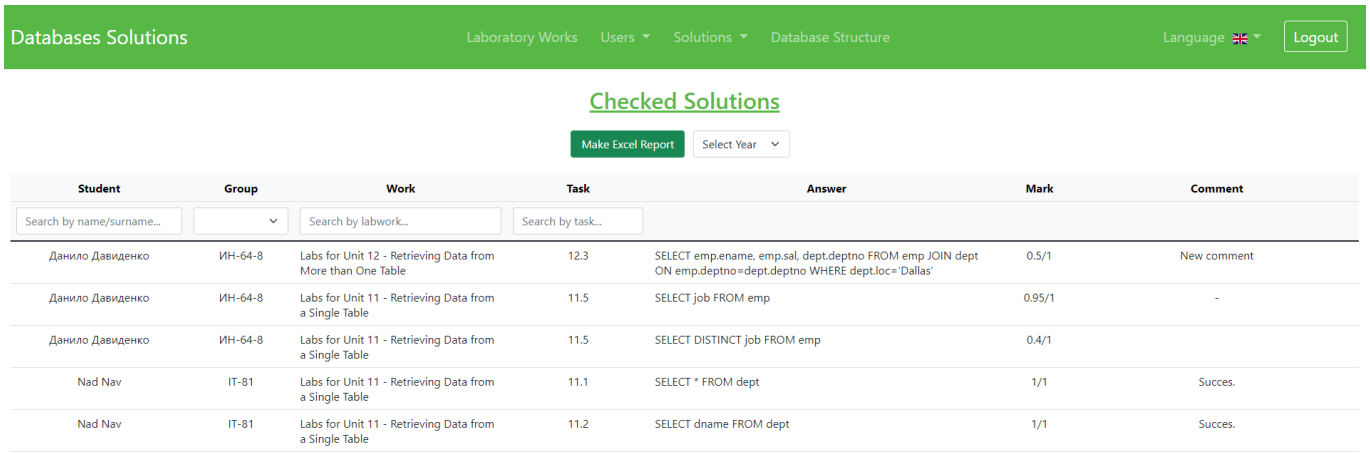


Рисунок 4.35 – Сторінка з перевіреними відповідями студентів

Для перегляду правильних відповідей до завдань передбачена окрема сторінка, на якій відображається відповідна таблиця (рис. 4.36).

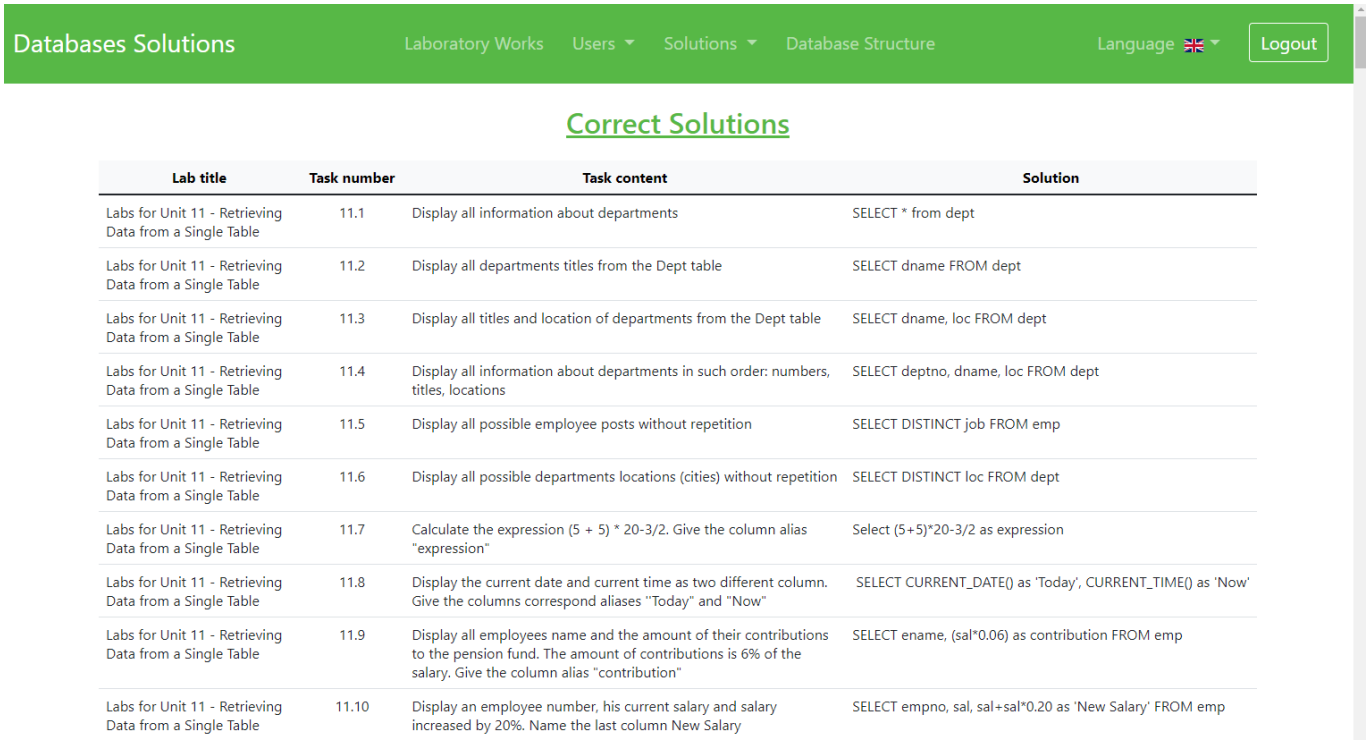


Рисунок 4.36 – Сторінка з правильними відповідями до завдань

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи магістра було проведено реінжиніринг інформаційної системи автоматизованої перевірки лабораторних робіт з вибірки даних.

Визначено актуальність проблеми, головну мету та задачі проекту. Оглянуто актуальні літературні джерела за означеними темами: реінжиніринг ПЗ, синтаксичний аналіз SQL запитів та проектування бази знань для підсистеми перевірки завдань. Проведено детальний аналіз схожих за тематикою та функціоналом програмних продуктів, і сформовано порівняльну таблицю цих рішень за визначеними характеристиками

За для досягнення поставленої мети було проведено планування робіт проекту. Ідентифіковано ідею проекту, проведено деталізацію мети за методологією SMART, планування змісту структури робі у вигляді WBS та OBS, побудовано діаграму Ґанта і визначено ризики проекту (додаток А). Обрано наступні засоби реалізації: для серверної частини web-застосунку було обрано мову програмування PHP 7.2 та систему керування базами даних MySQL 8.0, а для реалізації клієнтської частини використовується мова розмітки HTML5, таблиці стилів CSS3, CSS-фреймворк Bootstrap 5.1 та мова програмування JavaScript за стандартом ES8.

Створено модель функціонування головного бізнес-процесу (Організація виконання та оцінювання завдань з лабораторних робіт) у нотації IDEF0, а саме: її контекстну діаграму, та діаграми декомпозиції першого і другого рівня. Побудовано модель варіантів використання системи, спроектовано моделі бази даних системи: концептуальну і даталогічну, модель бази даних для бази знань з перевірки та оцінювання наданих відповідей, а також модель навчальної бази даних.

Визначено та впроваджено найбільш вдалий архітектурний шаблон програмного забезпечення – MVC. Згідно цього шаблону було модернізовано програмний код web-застосунку: створено окремі моделі для кожної сутності бази даних системи, контролери та вигляди для кожної функціональної сторінки системи.

Реалізовано модуль автоматизованої перевірки відповідей надісланих студентами, який передбачає перевірку правильності синтаксису та семантики відповіді (SQL запиту), та відображення підказок за результатами перевірки, які допомагають визначити причину помилки. Розширено функціонал для роботи з інформацією про користувачів. Значно покращено ефективність роботи з користувальницьким інтерфейсом: спрощено навігацію між завданнями однієї роботи, до ряду таблиць додано фільтри за встановленими атрибутами та можливість сортування записів, виправлено проблему з перевантаженням елементами інтерфейсу окремих сторінок системи.

Система впроваджена в навчальний процес на кафедрі інформаційних технологій факультету електроніки та інформаціних технологій. Акт впровадження наведено у додатку Б.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Da Mota Moura A. M. Awareness Driven Software Reengineering. *IEEE 25th International Requirements Engineering Conference (RE)* : матеріали конф., м. Лісабон, 4-8 вересня 2017. С. 550-555.

2 Великодний С. С., Тимофєєва О. С. Реінжиніринг програмного забезпечення інформаційних систем : монографія. Одеса : Видавничий дім «Гельветика», 2020. 160 с.

3 Великодний С., Тимофєєва О. Парадигма подання лінгвістичного забезпечення за допомогою породжувальних граматик. *Автоматизація технологій і бізнес-процесів*. 2017. №9. С. 46-50.

4 Kehe Wu. Design and implementation of a general SQL parser. *2021 4th International Conference on Computer Information Science and Application Technology* : матеріали конф., м. Ланьчжоу, 30 липня – 1 серпня 2021. 6 с.

5 Погромська Г. С. Синтаксичний аналізатор мови SQL. *Збірник наукових праць з актуальних проблем економічних наук*. 2017. 21-22 квітня. С. 43-46.

6 Vitkus D., Steckevicius Z., Goranin N., Kalibatiene D., Cenys A. Automated Expert System Knowledge Base Development Method for Information Security Risk Analysis. *International Journal of Computers Communications & Control*. 2019 Vol. 14, № 6, P. 743-758.

7 Шаров С. В., Печерський Р. В., Козлов С. М., Шарова Т. М. Розробка навчальної експертної системи для компаративного аналізу художніх творів. Вісник КрНУ імені Михайла Остроградського. Професійна освіта. 2020. С. 34-41

8 Інтерактивний онлайн тренажер мови SQL SQLCourse // 2021 URL: https://www.sqlcourse.com/ (дата звернення: 02.11.2021).

9 Інтерактивний онлайн тренажер мови SQL SQLZoo // 2021 URL: https://sqlzoo.net/wiki/SQL_Tutorial (дата звернення: 02.11.2021).

10 Платформа для оцінювання навичок спеціалістів HackerRank // 2021 URL: https://www.hackerrank.com/dashboard (дата звернення: 03.11.2021).

11 Інтерактивний онлайн тренажер мови SQL SQLBolt // 2021 URL: https://sqlbolt.com/ (дата звернення: 03.11.2021).

12 Web-додаток оцінювання лабораторних робіт з дисципліни «Організація баз даних та знань» // 2021 URL: http://db-sql-practics.zzz.com.ua/index.php (дата звернення: 04.11.2021).

13 Scott Berkun. Making Things Happen: Theory in Practice. O'Reilly Media Revised edition, 2013. 392 p.

14 Stanley E. Portny. Project Management For Dummies. For Dummies 4th edition, 2013. 408 p.

15 Grady Booch, James Rumbaugh, Ivar Jacobson. The Unified Modeling Language Usere Guide 2nd Edition. Addison-Wesley. 2006. 493 p.

16 Разработка функциональной модели // Методология IDEF0 URL: https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema6/tema6_2 (дата звернення: 14.11.2021).

17 Концепутальная модель базы данных // Диаграмма ER в СУБД URL: https://coderlessons.com/tutorials/bazy-dannykh/osnovy-subd/5-diagramma-er-v-subd (дата звернення: 15.11.2021).

18 Модель вариантов использования // Диаграммы вариантов использования URL: https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema12/tema12_2 (дата звернення: 15.11.2021).

19 Миграция данных // Процесс миграции, типы и золотые правила URL: https://asu-analitika.ru/migracija-dannyh-process-tipy-i-zolotye-pravila/ (дата звернення: 24.11.2021).

20 Sikha Bagui, Richard Earp, Database Design Using Entity-Relationship Diagrams (Foundations of Database Design) 2nd Edition. Auerbach Publications. 2011. 371 p.

21 Jaakkola H., Thalheim B., Kiyoki Y., Yoshida N., Dahanayake, A., Huiskonen J. Information Modelling and Knowledge Bases XXXI. IOS Press. 2019. 566 p.

22 Marchenko A., Kuzikov B., Drozdenko O. Methodological instructions for laboratory lessons on the database querying on the discipline “Organization of Databases and Knowledge Bases”: for students of specialties 122 “Computer Science” and 151

“Automation and Computer-Integrated Technologies” of full-time and part-time training. Sumy: Sumy State University. 2021. 24 p.

23 Jemie Chan SQL: Learn SQL (using MySQL) in One Day and Learn It Well. SQL for Beginners with Hands-on Project. LCF Publishing. 2018. 164 p.

24 Документація PHP // Руководство по PHP URL: https://www.php.net/manual/ru/index.php (дата звернення: 09.12.2021).

25 Christopher Pitt Pro PHP 8 MVC: Model View Controller Architecture-Driven Application Development 2nd Edition. Apress. 2021. 388 p.

26 Kevin Tatroe, Peter MacIntyre Programming PHP: Creating Dynamic Web Pages 4th Edition. O'Reilly Media. 2020. 544 p.

27 Luke Welling, Laura Thomson PHP and MySQL Web Development (Developer's Library) Revised Edition. Addison-Wesley. 2016. 688 p.

28 Marijn Haverbeke Eloquent JavaScript: A Modern Introduction to Programming 2nd Edition. No Starch Press. 2015. 472 p.

29 Дакетт Дж. Javascript и jQuery. Интерактивная веб-разработка. ЭКСМО. 2017. 640 с.

30 Jennifer Robbins Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics 4th Edition. O'Reilly Media. 2012. 624 p.

ДОДАТОК А.
ПЛАНУВАННЯ РОБІТ

А.1 Ідентифікація ідеї ІТ-проекту

Інформаційна система для автоматизованої перевірки лабораторних робіт з вибірки даних є дуже вдалим та зручним рішенням, що підвищує продуктивність роботи студентів і викладача. Проте, наявна реалізація цієї системи має ряд значних недоліків, які заважають ефективно користуватися нею.

Основними проблемами, з якими може стикнутися користувач системи, є застарілий і «недружній» інтерфейс, а також низька точність при перевірці відповіді та відсутність пояснювальної інформації при використанні модуля для оцінюванні лабораторних робіт. Студентам, що виконують завдання з лабораторної роботи, має стати у нагоді допоміжний функціонал, що дозволяє натисканням однієї кнопки додавати найбільш уживані конструкції мови SQL у поле для відповіді. Після підтвердження своєї відповіді до завдання вони мають бачити результат її перевірки, а для хибної відповіді також і пояснення результату. Крім того, студенти повинні мати змогу швидко здійснювати навігацію між завданнями лабораторної роботи. Викладачу, що виступає адміністратором системи, повинно бути зручно проводити CRUD-операції з інформацією до лабораторних робіт та користувачів системи. Повинна бути мінімізована необхідність власноруч перевіряти, надіслані студентами, відповіді, і залишена можливість підтвердження правильності автоматизованої перевірки цих відповідей.

Таким чином, на основі зазначених потреб користувачів сформувалася ідея проекту, метою якого є осучаснення архітектури програмного забезпечення ІС для дисципліни «Організація баз даних та знань», покращення інтерфейсу системи, і вдосконалення програмного модуля автоматизованої перевірки завдань з лабораторних робіт.

А.2 Деталізація мети проекту методом SMART

Продуктом дипломного проекту є модернізована ІС автоматизованої перевірки лабораторних робіт з вибірки даних. Результати деталізації мети методом SMART розміщені у таблиці А.1.

Таблиця А.1 – Деталізація мети методом SMART

Specific (конкретна)	Модернізувати архітектуру програмного забезпечення ІС, покращити інтерфейс та вдосконалити модуль автоматизованої перевірки завдань
Measurable (вимірювана)	Щонайменше вдвічі підвищити ефективність роботи з системою (оцінка проводиться користувачами системи)
Achievable (досяжна)	Модернізувати архітектуру ПЗ відповідно до обраного архітектурного шаблону
Relevant (актуальна)	Осучаснити інтерфейс та впровадити підсистему перевірки завдань для оцінювання завдань з лабораторних робіт
Time-bound (обмежена у часі)	Провести модернізацію системи протягом двох місяців

А.3 Планування змісту структури робіт ІТ-проекту

Work Breakdown Structure (WBS) розроблюється для загального розуміння змісту проекту. Це ієрархічний опис роботи, яка має бути виконана для досягнення результатів проекту. Кожен низхідний рівень WBS представляє собою більш докладний опис результатів проекту. Серед основних причин використання цього інструменту: сприяє розробці календарного плану, забезпечую візуалізацію всього

об’єму робіт, можна використовувати для відображення й розподілу відповідальності та обов’язків, гарантує, що жодна важлива задача не буде пропущена та може допомогти з розподілом ресурсів. WBS для реінжинірингу ІС наведено на рисунку А.1.

На основі отриманої декомпозиції робіт проекту відбувається розподіл робіт між членами команди, і цей розподіл зазначається в організаційній структурі робіт. Organizational Breakdown Structure (OBS), що будується аналогічно WBS називається функціональною. У цьому випадку на найвищому рівні, як єдиний елемент, зазначається організаційна структура, тобто виконавці проекту, а потім вони розподіляються по елементарних роботах, що зазначені у WBS. Функціональну OBS для реінжинірингу ІС наведено на рисунку А.2.

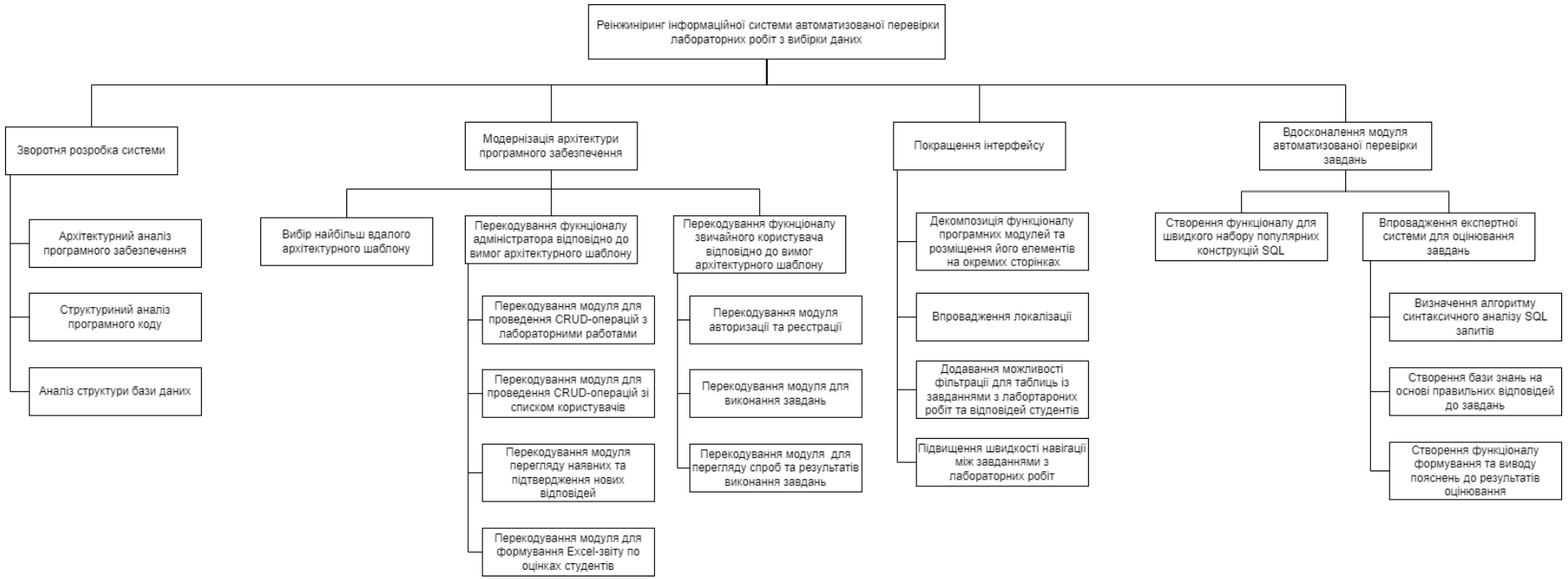


Рисунок А.1 – Таблиця WBS

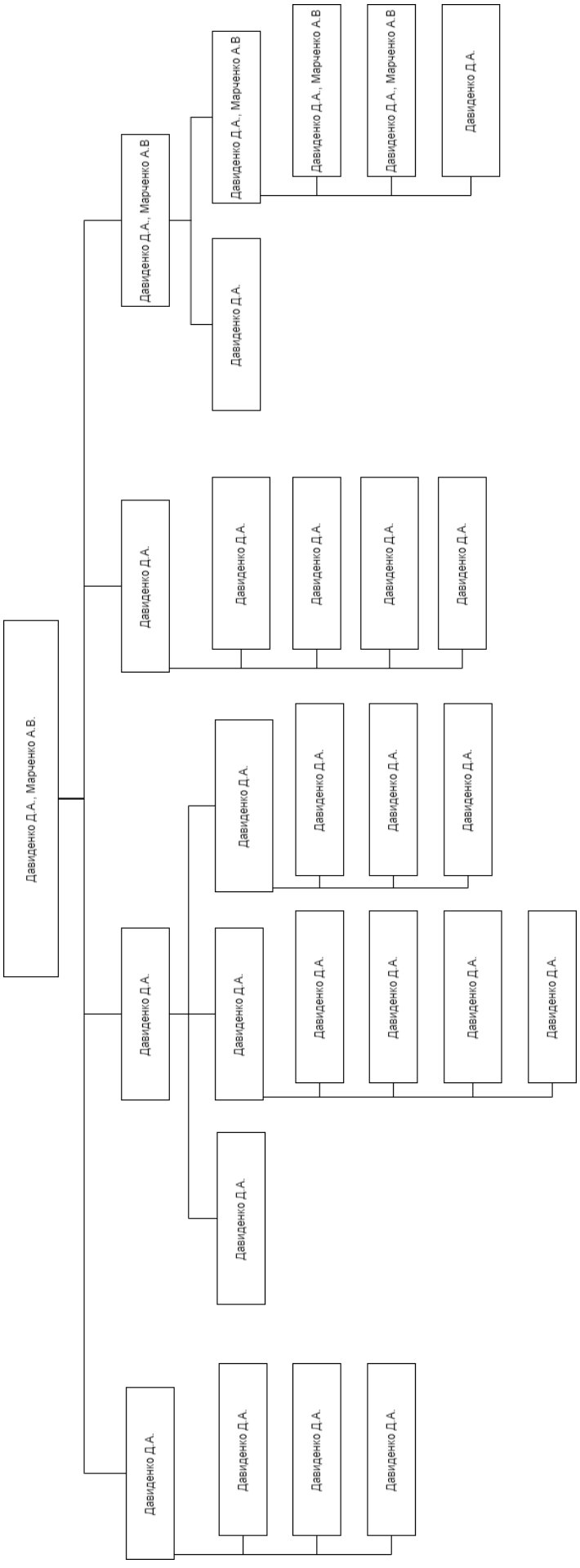
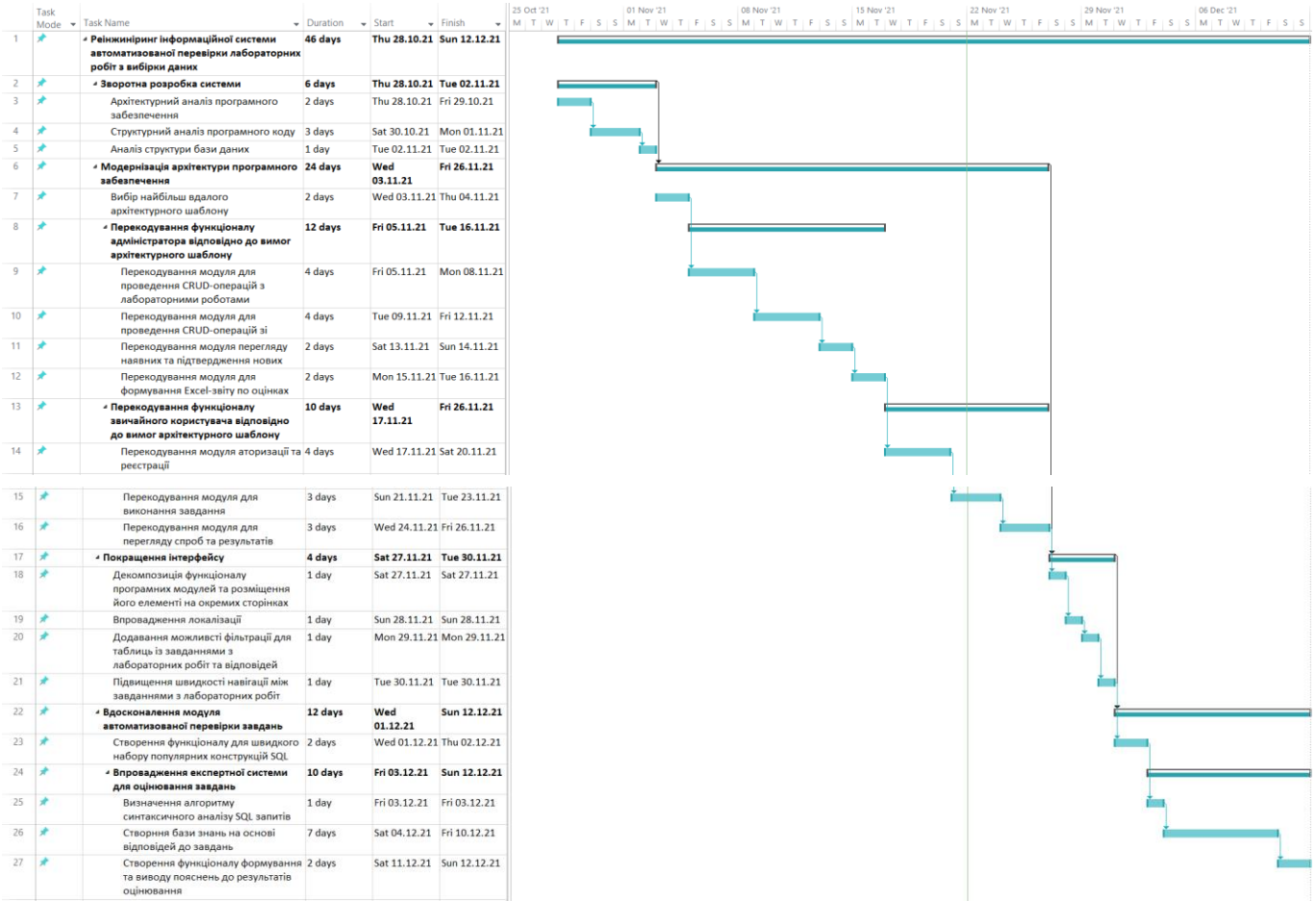


Рисунок А.2 – Таблиця OBS

А.4 Побудована календарного графіку виконання ІТ-проекту

Формування графіку виконання робіт по проекту відбувається за допомогою побудови діаграми Ґанта. Ця діаграма представляє собою інструмент керування проектами, що допомагає у плануванні та створенні розкладу проектів будь-якого розміру. Терміни та завдання проекту перетворюються на горизонтальну лінійчату діаграму, на якій відображаються дати початку та закінчення, залежності й зв’язки між задачами, а також виконавці цих задач (якщо проект виконує команда). На рисунку А.3 наведено діаграму Ґанта побудовану з використанням додатку Microsoft Project.



	Task Mode	Task Name	Duration	Start	Finish
1		**Реінжиніринг інформаційної системи автоматизованої перевірки лабораторних робіт з вибірки даних**	**46 days**	**Thu 28.10.21**	**Sun 12.12.21**
2		**Зворотна розробка системи**	**6 days**	**Thu 28.10.21**	**Tue 02.11.21**
3		Архітектурний аналіз програмного забезпечення	2 days	Thu 28.10.21	Fri 29.10.21
4		Структурний аналіз програмного коду	3 days	Sat 30.10.21	Mon 01.11.21
5		Аналіз структури бази даних	1 day	Tue 02.11.21	Tue 02.11.21
6		**Модернізація архітектури програмного забезпечення**	**24 days**	**Wed 03.11.21**	**Fri 26.11.21**
7		Вибір найбільш вдалого архітектурного шаблону	2 days	Wed 03.11.21	Thu 04.11.21
8		**Перекодування функціоналу адміністратора відповідно до вимог архітектурного шаблону**	**12 days**	**Fri 05.11.21**	**Tue 16.11.21**
9		Перекодування модуля для проведення CRUD-операцій з лабораторними роботами	4 days	Fri 05.11.21	Mon 08.11.21
10		Перекодування модуля для проведення CRUD-операцій зі	4 days	Tue 09.11.21	Fri 12.11.21
11		Перекодування модуля перегляду наявних та підтвердження нових	2 days	Sat 13.11.21	Sun 14.11.21
12		Перекодування модуля для формування Excel-звіту по оцінках	2 days	Mon 15.11.21	Tue 16.11.21
13		**Перекодування функціоналу звичайного користувача відповідно до вимог архітектурного шаблону**	**10 days**	**Wed 17.11.21**	**Fri 26.11.21**
14		Перекодування модуля аторизації та реєстрації	4 days	Wed 17.11.21	Sat 20.11.21
15		Перекодування модуля для виконання завдання	3 days	Sun 21.11.21	Tue 23.11.21
16		Перекодування модуля для перегляду спроб та результатів	3 days	Wed 24.11.21	Fri 26.11.21
17		**Покращення інтерфейсу**	**4 days**	**Sat 27.11.21**	**Tue 30.11.21**
18		Декомпозиція функціоналу програмних модулей та розміщення його елементі на окремих сторінках	1 day	Sat 27.11.21	Sat 27.11.21
19		Впровадження локалізації	1 day	Sun 28.11.21	Sun 28.11.21
20		Додавання можливсті фільтрації для таблиць із завданнями з лабораторних робіт та відповідей	1 day	Mon 29.11.21	Mon 29.11.21
21		Підвищення швидкості навігації між завданнями з лабораторних робіт	1 day	Tue 30.11.21	Tue 30.11.21
22		**Вдосконалення модуля автоматизованої перевірки завдань**	**12 days**	**Wed 01.12.21**	**Sun 12.12.21**
23		Створення функціоналу для швидкого набору популярних конструкцій SQL	2 days	Wed 01.12.21	Thu 02.12.21
24		**Впровадження експертної системи для оцінювання завдань**	**10 days**	**Fri 03.12.21**	**Sun 12.12.21**
25		Визначення алгоритму синтаксичного аналізу SQL запитів	1 day	Fri 03.12.21	Fri 03.12.21
26		Створення бази знань на основі відповідей до завдань	7 days	Sat 04.12.21	Fri 10.12.21
27		Створення функціоналу формування та виводу пояснень до результатів оцінювання	2 days	Sat 11.12.21	Sun 12.12.21

Рисунок А.3 – Діаграма Ґанта

А.5 Планування ризиків проекту

Управління ризиками проекту – це процес виявлення, аналізу та реагування на будь-який ризик, що виникає протягом життєвого циклу проекту. Крім того, управління ризиками є частиною процесу планування, що необхідно для своєчасного з’ясування можливих ризиків, і контролю цих ризиків, якщо вони дійсно виникають. Ризик – це все, що потенційно може вплинути на терміни, продуктивність або бюджет проекту. Процес ідентифікації ризиків передбачає встановлення та документування характеристик ризиків, які можуть вплинути на проект. Цей процес є ітеративним, адже з розвитком проекту можуть з’являтися нові ризики, що потребуватимуть ідентифікації та реагування.

Перелічимо можливі ризики проекту, визначимо відносну критичність, а також можливі способи їх уникнення. Введемо умовну шкалу рівнів критичності ризиків (табл. А.2):

Таблиця А.2 – Рівні критичності ризиків

1	Незначний
2	Низький
3	Середній
4	Високий
5	Критичний

Ризики проекту, визначені в результаті ідентифікації, а також їх потенційний вплив та ймовірність виникнення представлені у таблиці А.3.

Таблиця А.3 – Ризики проекту

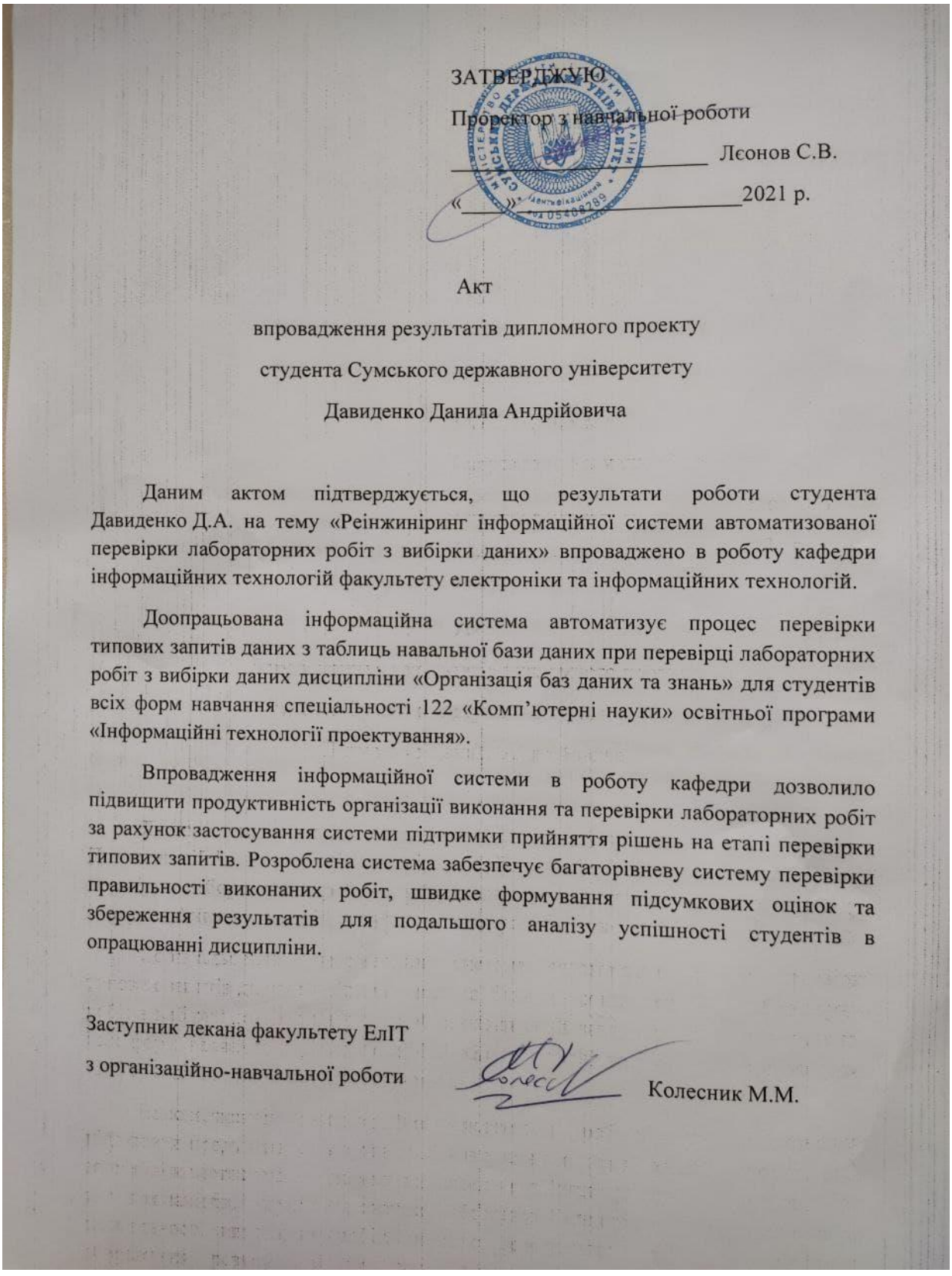
№	Опис	Ймовірність	Вплив
1	Втрата актуальності проекту	1	4
2	Зміна вимог на пізніх етапах розробки	2	4
3	Незаплановані роботи, що потребують обов’язкового виконання	2	3
4	Вимушене зміщення термінів виконання завдань або паралельне виконання цих завдань	3	3
5	Нездійснені або важкі задачі	3	5
6	Пошкодження або відсутність обладнання	1	5
7	Непрацездатність розробника	1	5
8	Низьке залучення замовника	2	4

Тепер, складемо матрицю залежності втрат від ймовірності виникнення кожного із зазначених ризиків (табл. А.4).

Таблиця А.4 – Матриця «Ймовірність – Втрати»

Ймовірність\Втрати	1	2	3	4	5
1				1	6,7
2			3	2,8	
3			4		5
4					
5					

ДОДАТОК Б.
АКТ ВПРОВАДЖЕННЯ



ЗАТВЕРДЖУЮ
Проректор з навчальної роботи
_____ Лєонов С.В.
«____»_____2021 р.

Акт

впровадження результатів дипломного проекту

студента Сумського державного університету

Давиденко Данила Андрійовича

Даним актом підтверджується, що результати роботи студента Давиденко Д.А. на тему «Реінжиніринг інформаційної системи автоматизованої перевірки лабораторних робіт з вибірки даних» впроваджено в роботу кафедри інформаційних технологій факультету електроніки та інформаційних технологій.

Доопрацьована інформаційна система автоматизує процес перевірки типових запитів даних з таблиць навальної бази даних при перевірці лабораторних робіт з вибірки даних дисципліни «Організація баз даних та знань» для студентів всіх форм навчання спеціальності 122 «Комп'ютерні науки» освітньої програми «Інформаційні технології проектування».

Впровадження інформаційної системи в роботу кафедри дозволило підвищити продуктивність організації виконання та перевірки лабораторних робіт за рахунок застосування системи підтримки прийняття рішень на етапі перевірки типових запитів. Розроблена система забезпечує багаторівневу систему перевірки правильності виконаних робіт, швидке формування підсумкових оцінок та збереження результатів для подальшого аналізу успішності студентів в опрацюванні дисципліни.

Заступник декана факультету ЕлІТ
з організаційно-навчальної роботи Колесник М.М.

ДОДАТОК В.
ЛІСТИНГ ФАЙЛІВ ПРОГРАМНОГО КОДУ

Лістинг основних файлів керування роботою додатку:

Core.php.

```
<?php
   class Core {
      protected $currentController = 'Users_Auth';
      protected $currentMethod = 'login';
      protected $params = [];

      public function __construct() {
         $url = $this->getUrl();
         // Getting controller name as a first array element.
         if($url)
         {
            $controller = ucwords($url[0], "_");
            if(file_exists('../app/controllers/' . $controller . '.php')) {
               $this->currentController = $controller;
               unset($url[0]);
            }
         }

         require '../app/controllers/' . $this->currentController . '.php';
         $this->currentController = new $this->currentController;
         // Getting method name by iterating over remaining array elements and searching for existing name.
         // Strict naming rules for files and controller`s methods!
         // Array elements remaining after searching for method become parameters of the method.
         if($url)
         {
            $urlCount = count($url);
            for($i = 1; $i <= $urlCount; $i++) {
               $method = $url[$i];
               if(method_exists($this->currentController, $method)) {
                  $this->currentMethod = $method;
                  unset($url[$i]);
                  break;
               }
```

```
            unset($url[$i]);
        }

        $this->params = array_values($url);
    }
    call_user_func_array([$this->currentController, $this->currentMethod], $this->params);
}

public function getUrl() {
    if(isset($_GET['url'])) {
        $url = rtrim($_GET['url'], '/');
        $url = filter_var($url, FILTER_SANITIZE_URL);
        $url = explode('/', $url);
        return $url;
    }
}
}
```

Controller.php.

```
<?php
class Controller {
    public function model($model) {
        require '../app/models/' . $model . '.php';
        return new $model();
    }

    public function view($view, $data = []) {
        if(file_exists('../app/views/' . $view . '.php')) {
            require '../app/views/' . $view . '.php';
        } else {
            die("No view.");
        }
    }
}
```

Database.php.

```
<?php
class Database {
    private $dbHost = DB_HOST;
    private $dbName = DB_NAME;
```

```
private $dbUser = DB_USER;
private $dbPass = DB_PASS;

private $statement;
private $dbHandler;
private $error;

public function __construct() {
    $connectionString = 'mysql:host=' . $this->dbHost . ';dbname=' . $this->dbName;
    $options = array(
        PDO::ATTR_PERSISTENT => true,
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION
    );

    try {
        $this->dbHandler = new PDO(
            $connectionString,
            $this->dbUser,
            $this->dbPass,
            $options
        );
    } catch(PDOException $ex) {
        $this->error = $ex->getMessage();
    }
}

public function setQuery($sql) {
    $this->statement = $this->dbHandler->prepare($sql);
}

public function bindQueryParameter($param, $value, $type = null) {
    switch(is_null($type)) {
        case is_bool($value):
            $type = PDO::PARAM_BOOL;
            break;
        case is_int($value):
            $type = PDO::PARAM_INT;
            break;
        case is_null($value):
            $type = PDO::PARAM_NULL;
            break;
        default:
```

```
            $type = PDO::PARAM_STR;
        }

        $this->statement->bindValue($param, $value, $type);
    }

    public function executeQuery() {
        return $this->statement->execute();
    }

    public function getSingleResult() {
        $this->executeQuery();
        return $this->statement->fetch(PDO::FETCH_OBJ);
    }

    public function getArrayResult() {
        $this->executeQuery();
        return $this->statement->fetchAll(PDO::FETCH_OBJ);
    }

    public function getAffectedRowsCount() {
        return $this->statement->rowCount();
    }
}
```

EducationalDatabase.php.

```
<?php
    class EducationalDatabase {
        private $dbHost = DB_HOST_EDC;
        private $dbName = DB_NAME_EDC;
        private $dbUser = DB_USER_EDC;
        private $dbPass = DB_PASS_EDC;

        private $dbHandler;
        private $error;

        public function __construct() {
            $connectionString = 'mysql:host=' . $this->dbHost . ';dbname=' . $this->dbName;
            $options = array(
                PDO::ATTR_EMULATE_PREPARES => false,
            );
```

```
        try {
            $this->dbHandler = new PDO(
                $connectionString,
                $this->dbUser,
                $this->dbPass,
                $options
            );
        } catch(PDOException $ex) {
            $this->error = $ex->getMessage();
        }
    }

    public function tryQuery($sql) {
        $queryRes = $this->dbHandler->prepare($sql);
        if($queryRes) {
            return true;
        } else {
            return $this->dbHandler->errorInfo();
        }
    }
}
```

Лістинг основних файлів моделей:

Group.php.

```
<?php
  class Group {
    private $db;

    public function __construct() {
        $this->db = new Database;
    }

    public function getAllGroups() {
        $this->db->setQuery('SELECT * FROM `groups`');

        $resultRows = $this->db->getArrayResult();
        return $resultRows;
    }

    public function getGroupById($id) {
```

```
        $this->db->setQuery('SELECT *
                    FROM `groups`
                    WHERE group_id = :group_id');

        $this->db->bindQueryParameter(':group_id', $id);

        $resultRow = $this->db->getSingleResult();
        return $resultRow;
    }

    public function getGroupByNumber($number) {
        $this->db->setQuery('SELECT *
                    FROM `groups`
                    WHERE number = :number');

        $this->db->bindQueryParameter(':number', $number);

        $resultRow = $this->db->getSingleResult();
        return $resultRow;
    }

    public function getGroupsWithSolutionsByLabYear($year) {
        $this->db->setQuery('SELECT g.*
                    FROM `groups` AS g
                    JOIN `users` AS u ON g.group_id = u.group_id
                    JOIN `student_solutions` AS ss ON u.user_id = ss.user_id
                    JOIN `laboratory_works` AS lw ON ss.lab_id = lw.lab_id AND YEAR(lw.end_date) = :year
                    GROUP BY g.group_id');

        $this->db->bindQueryParameter(':year', $year);

        $resultRows = $this->db->getArrayResult();
        return $resultRows;
    }
}
```

LaboratoryWork.php.

```
<?php
    class LaboratoryWork {
        private $db;
```

```
public function __construct() {
   $this->db = new Database;
}

public function getLaboratoryWorks() {
   $this->db->setQuery('SELECT *
                  FROM `laboratory_works`
                  ORDER BY lab_id');

   $resultRows = $this->db->getArrayResult();
   return $resultRows;
}

public function getLaboratoryWorkById($lab_id) {
   $this->db->setQuery('SELECT *
                  FROM `laboratory_works`
                  WHERE lab_id = :lab_id');

   $this->db->bindQueryParameter(':lab_id', $lab_id);

   $resultRow = $this->db->getSingleResult();
   return $resultRow;
}

public function getLaboratoryWorksByYear($year) {
   $this->db->setQuery('SELECT *
                  FROM `laboratory_works`
                  WHERE YEAR(end_date) = :year
                  ORDER BY end_date');

   $this->db->bindQueryParameter(':year', $year);

   $resultRow = $this->db->getSingleResult();
   return $resultRow;
}

public function insert($data) {
   $this->db->setQuery('INSERT INTO `laboratory_works` (title_en, title_ua, start_date, end_date, max_mark)
                        VALUES (:title_en, :title_ua, :start_date, :end_date, :max_mark)');

   $this->db->bindQueryParameter(':title_en', $data['titleEN']);
   $this->db->bindQueryParameter(':title_ua', $data['titleUA']);
```

```
        $this->db->bindQueryParameter(':start_date', $data['startDate']);
        $this->db->bindQueryParameter(':end_date', $data['endDate']);
        $this->db->bindQueryParameter(':max_mark', floatval($data['maxMark']));

        return $this->db->executeQuery();
    }

    public function update($data) {
        $this->db->setQuery('UPDATE `laboratory_works`
                    SET title_en = :title_en,
                        title_ua = :title_ua,
                        start_date = :start_date,
                        end_date = :end_date,
                        max_mark = :max_mark
                    WHERE lab_id = :lab_id');

        $this->db->bindQueryParameter(':title_en', $data['titleEN']);
        $this->db->bindQueryParameter(':title_ua', $data['titleUA']);
        $this->db->bindQueryParameter(':start_date', $data['startDate']);
        $this->db->bindQueryParameter(':end_date', $data['endDate']);
        $this->db->bindQueryParameter(':max_mark', floatval($data['maxMark']));
        $this->db->bindQueryParameter(':lab_id', $data['lab_id']);

        return $this->db->executeQuery();
    }

    public function delete($id) {
        $this->db->setQuery('DELETE
                    FROM `laboratory_works`
                    WHERE lab_id = :lab_id');

        $this->db->bindQueryParameter(':lab_id', $id);

        return $this->db->executeQuery();
    }
}
```

Task.php.

```
<?php
  class Task {
    private $db;
```

```
public function __construct() {
   $this->db = new Database;
}

public function getTaskById($id) {
   $this->db->setQuery('SELECT *
                  FROM `tasks`
                  WHERE task_id = :task_id');

   $this->db->bindQueryParameter(':task_id', $id);

   $resultRow = $this->db->getSingleResult();
   return $resultRow;
}

public function getTasksByLab($lab_id) {
   $this->db->setQuery('SELECT *
                  FROM `tasks`
                  WHERE lab_id = :lab_id
                  ORDER BY task_id');

   $this->db->bindQueryParameter(':lab_id', $lab_id);

   $resultRows = $this->db->getArrayResult();
   return $resultRows;
}

public function getNextAndPreviousTasksIds($lab_id, $task_id) {
   $this->db->setQuery('SELECT COALESCE((SELECT task_id
                              FROM `tasks`
                              WHERE lab_id = :lab_id AND task_id < :task_id
                              ORDER BY task_id DESC
                              LIMIT 1), 0) AS nptask_id
                  UNION
                  SELECT COALESCE((SELECT task_id
                              FROM `tasks`
                              WHERE lab_id = :lab_id AND task_id > :task_id
                              ORDER BY task_id
                              LIMIT 1), 0) AS nptask_id');

   $this->db->bindQueryParameter(':lab_id', $lab_id);
```

```
    $this->db->bindQueryParameter(':task_id', $task_id);

    $resultRows = $this->db->getArrayResult();
    return $resultRows;
}

public function insert($data) {
    $this->db->setQuery('INSERT INTO `tasks` (number, exercise_en, exercise_ua, difficulty, attempts, lab_id)
                        VALUES (:number, :exercise_en, :exercise_ua, :difficulty, :attempts, :lab_id)');

    $this->db->bindQueryParameter(':number', $data['number']);
    $this->db->bindQueryParameter(':exercise_en', $data['exerciseEN']);
    $this->db->bindQueryParameter(':exercise_ua', $data['exerciseUA']);
    $this->db->bindQueryParameter(':difficulty', $data['difficulty']);
    $this->db->bindQueryParameter(':attempts', $data['attempts']);
    $this->db->bindQueryParameter(':lab_id', $data['lab_id']);

    return $this->db->executeQuery();
}

public function update($data) {
    $this->db->setQuery('UPDATE `tasks`
                    SET number = :number,
                        exercise_en = :exercise_en,
                        exercise_ua = :exercise_ua,
                        difficulty = :difficulty,
                        attempts = :attempts
                    WHERE task_id = :task_id');

    $this->db->bindQueryParameter(':number', $data['number']);
    $this->db->bindQueryParameter(':exercise_en', $data['exerciseEN']);
    $this->db->bindQueryParameter(':exercise_ua', $data['exerciseUA']);
    $this->db->bindQueryParameter(':difficulty', $data['difficulty']);
    $this->db->bindQueryParameter(':attempts', $data['attempts']);
    $this->db->bindQueryParameter(':task_id', $data['task_id']);

    return $this->db->executeQuery();
}

public function delete($id) {
    $this->db->setQuery('DELETE
                    FROM `tasks`
```

```
                WHERE task_id = :task_id');

        $this->db->bindQueryParameter(':task_id', $id);

        return $this->db->executeQuery();
    }
}
```

Solution.php.

```
<?php
class Solution {
    private $db;

    public function __construct() {
        $this->db = new Database;
    }

    public function getNewStudentsSolutions() {
        $this->db->setQuery('SELECT ss.s_solution_id AS s_solution_id, ss.answer AS answer, ss.mark AS mark, ss.comment
AS comment, t.number AS task_number,
                t.difficulty AS difficulty, lw.title_en AS lab_title, u.name AS name, u.surname AS surname, g.number
AS group_number
                FROM `student_solutions` AS ss
                LEFT JOIN `tasks` AS t ON ss.task_id = t.task_id
                LEFT JOIN `laboratory_works` AS lw ON ss.lab_id = lw.lab_id
                LEFT JOIN `users` AS u ON ss.user_id = u.user_id
                LEFT JOIN `groups` AS g ON u.group_id = g.group_id
                WHERE ss.confirmed = :confirmed
                ORDER BY answer_date ASC');

        $this->db->bindQueryParameter(':confirmed', 0);

        $resultRows = $this->db->getArrayResult();
        return $resultRows;
    }

    public function getNewStudentsSolutionById($id) {
        $this->db->setQuery('SELECT ss.s_solution_id AS s_solution_id, ss.answer AS answer, ss.answer_date AS
answer_date, ss.mark AS mark,
                ss.comment AS comment, t.number AS task_number, t.exercise_en AS exercise, t.difficulty AS
difficulty,
```

```
                lw.title_en AS lab_title, u.name AS name, u.surname AS surname, g.number AS group_number
            FROM `student_solutions` AS ss
            LEFT JOIN `tasks` AS t ON ss.task_id = t.task_id
            LEFT JOIN `laboratory_works` AS lw ON ss.lab_id = lw.lab_id
            LEFT JOIN `users` AS u ON ss.user_id = u.user_id
            LEFT JOIN `groups` AS g ON u.group_id = g.group_id
            WHERE ss.s_solution_id = :solution_id AND ss.confirmed = :confirmed');

        $this->db->bindQueryParameter(':solution_id', $id);
        $this->db->bindQueryParameter(':confirmed', 0);

        $resultRow = $this->db->getSingleResult();
        return $resultRow;
    }

    public function getCheckedStudentsSolutions() {
        $this->db->setQuery('SELECT ss.answer AS answer, ss.mark AS mark, ss.comment AS comment, t.number AS
task_number, t.difficulty AS difficulty,
                lw.title_en AS lab_title, u.name AS name, u.surname AS surname, g.number AS group_number
            FROM `student_solutions` AS ss
            LEFT JOIN `tasks` AS t ON ss.task_id = t.task_id
            LEFT JOIN `laboratory_works` AS lw ON ss.lab_id = lw.lab_id
            LEFT JOIN `users` AS u ON ss.user_id = u.user_id
            LEFT JOIN `groups` AS g ON u.group_id = g.group_id
            WHERE ss.confirmed = :confirmed
            ORDER BY answer_date ASC');

        $this->db->bindQueryParameter(':confirmed', 1);

        $resultRows = $this->db->getArrayResult();
        return $resultRows;
    }

    public function getCorrectSolutions() {
        $this->db->setQuery('SELECT ts.*, t.number AS task_number, t.exercise_en AS task_content, lw.title_en AS lab_title
            FROM `teacher_solutions` AS ts
            LEFT JOIN `tasks` AS t ON ts.task_id = t.task_id
            LEFT JOIN `laboratory_works` AS lw ON ts.lab_id = lw.lab_id');

        $resultRows = $this->db->getArrayResult();
        return $resultRows;
    }
```

```
public function getCorrectSolutionByTaskId($task_id) {
    $this->db->setQuery('SELECT *
                FROM `teacher_solutions`
                WHERE task_id = :task_id');

    $this->db->bindQueryParameter(':task_id', $task_id);

    $resultRow = $this->db->getSingleResult();
    return $resultRow;
}

public function getStudentAttemptSolutions($task_id, $user_id) {
    $this->db->setQuery('SELECT ss.answer AS answer, ss.answer_date AS answer_date, ss.mark AS mark,
                    ss.comment AS comment, ss.confirmed AS confirmed
                FROM `student_solutions` AS ss
                LEFT JOIN `tasks` AS t ON ss.task_id = t.task_id
                LEFT JOIN `users` AS u ON ss.user_id = u.user_id
                WHERE t.task_id = :task_id AND ss.user_id = :user_id
                ORDER BY answer_date ASC');

    $this->db->bindQueryParameter(':task_id', $task_id);
    $this->db->bindQueryParameter(':user_id', $user_id);

    $resultRows = $this->db->getArrayResult();
    return $resultRows;
}

public function getStudentAttemptsOnTaskCount($task_id, $user_id) {
    $this->db->setQuery('SELECT COUNT(*) AS solutions_count
                FROM `student_solutions` AS ss
                LEFT JOIN `tasks` AS t ON ss.task_id = t.task_id
                WHERE t.task_id = :task_id AND ss.user_id = :user_id');

    $this->db->bindQueryParameter(':task_id', $task_id);
    $this->db->bindQueryParameter(':user_id', $user_id);

    $rowsCount = $this->db->getSingleResult();
    return $rowsCount;
}

public function getTasksAttemptsLeft($lab_id, $user_id) {
```

```
        $this->db->setQuery('SELECT t.attempts - COUNT(ss.task_id) AS attempts_left
                    FROM `student_solutions` AS ss
                    RIGHT JOIN `tasks` AS t ON ss.task_id = t.task_id
                    WHERE t.lab_id = :lab_id AND (ss.user_id = :user_id OR ss.user_id IS NULL)
                    GROUP BY t.task_id
                    ORDER BY t.task_id');

        $this->db->bindQueryParameter(':lab_id', $lab_id);
        $this->db->bindQueryParameter(':user_id', $user_id);

        $resultRows = $this->db->getArrayResult();
        return $resultRows;
    }

    public function getTasksBestScores($lab_id, $user_id) {
        $this->db->setQuery('SELECT COALESCE(MAX(ss.mark), "0") AS best_score
                    FROM `student_solutions` AS ss
                    RIGHT JOIN `tasks` AS t ON ss.task_id = t.task_id
                    WHERE t.lab_id = :lab_id AND (ss.user_id = :user_id OR ss.user_id IS NULL)
                    GROUP BY t.task_id
                    ORDER BY t.task_id');

        $this->db->bindQueryParameter(':lab_id', $lab_id);
        $this->db->bindQueryParameter(':user_id', $user_id);

        $resultRows = $this->db->getArrayResult();
        return $resultRows;
    }

    public function getLabsCurrentMarks($user_id) {
        $this->db->setQuery('SELECT COALESCE(ROUND(max_mark * (ss.sum_marks/t.sum_difficulty), 1), 0) AS
current_mark
                    FROM `laboratory_works` AS lw
                    LEFT JOIN (
                        SELECT lab_id, SUM(difficulty) AS sum_difficulty
                        FROM `tasks`
                        GROUP BY lab_id
                    ) AS t ON lw.lab_id = t.lab_id
                    LEFT JOIN (
                        SELECT lab_id, SUM(ROUND(mark, 2)) AS sum_marks
                        FROM `student_solutions`
                        WHERE confirmed = :confirmed AND user_id = :user_id
```

```
                    GROUP BY lab_id
                ) AS ss ON lw.lab_id = ss.lab_id
                ORDER BY lw.lab_id');

        $this->db->bindQueryParameter(':user_id', $user_id);
        $this->db->bindQueryParameter(':confirmed', 1);

        $resultRows = $this->db->getArrayResult();
        return $resultRows;
    }

    public function insert($data) {
        $this->db->setQuery('INSERT INTO `student_solutions` (answer, answer_date, user_id, lab_id, task_id, mark,
comment, confirmed)
                        VALUES (:answer, :answer_date, :user_id, :lab_id, :task_id, :mark, :comment, :confirmed)');

        $this->db->bindQueryParameter(':answer', $data['answer']);
        $this->db->bindQueryParameter(':answer_date', date('Y-m-d H:i:s'));
        $this->db->bindQueryParameter(':user_id', $data['user_id']);
        $this->db->bindQueryParameter(':lab_id', $data['lab_id']);
        $this->db->bindQueryParameter(':task_id', $data['task_id']);
        $this->db->bindQueryParameter(':mark', floatval($data['mark']));
        $this->db->bindQueryParameter(':comment', $data['comment']);
        $this->db->bindQueryParameter(':confirmed', 0);

        return $this->db->executeQuery();
    }

    public function confirm($id) {
        $this->db->setQuery('UPDATE `student_solutions`
                SET confirmed = :confirmed
                WHERE s_solution_id = :s_solution_id');

        $this->db->bindQueryParameter(':confirmed', 1);
        $this->db->bindQueryParameter(':s_solution_id', $id);

        return $this->db->executeQuery();
    }

    public function saveConfirm($data) {
        $this->db->setQuery('UPDATE `student_solutions`
                SET comment = :comment,
```

```
                    mark = :mark,
                    confirmed = :confirmed
                WHERE s_solution_id = :s_solution_id');

        $this->db->bindQueryParameter(':comment', $data['comment']);
        $this->db->bindQueryParameter(':mark', floatval($data['mark']));
        $this->db->bindQueryParameter(':confirmed', 1);
        $this->db->bindQueryParameter(':s_solution_id', $data['id']);

        return $this->db->executeQuery();
    }

    public function delete($id) {
        $this->db->setQuery('DELETE
                FROM `student_solutions`
                WHERE s_solution_id = :s_solution_id');

        $this->db->bindQueryParameter(':s_solution_id', $id);

        return $this->db->executeQuery();
    }
}
```

TaskCheck.php.

```
<?php
class TaskCheck {
    private $db;
    private $db_edc;

    public function __construct() {
        $this->db = new Database;
        $this->db_edc = new EducationalDatabase;
    }

    public function checkQuerySyntax($query) {
        return $this->db_edc->tryQuery($query . ";");
    }

    public function checkQuerySemantics($answerQuery, $correct_query_id) {
        $queryType = $this->getQueryType($correct_query_id);
        $queryComponents = $this->getQueryComponents($correct_query_id);
```

```
    $hints = [];

    switch(trim($queryType->definition)) {
        case "SELECT SIMPLE":
            $this->simpleSelectCheck($answerQuery, $queryComponents, $hints);
            break;
        case "SELECT JOIN":
            $this->joinSelectCheck($answerQuery, $queryComponents, $hints);
            break;
        case "CREATE":
            $this->createCheck($answerQuery, $queryComponents, $hints);
            break;
        case "PLAIN TEXT":
            $this->plainTextCheck($answerQuery, $queryComponents, $hints);
            break;
    }

    if(empty($hints)) {
        return true;
    } else {
        return $hints;
    }
}

private function getQueryType($query_id) {
    $this->db->setQuery('SELECT t.*
                    FROM `types` t
                    LEFT JOIN `teacher_solutions` ts ON t.type_id = ts.type_id
                    WHERE ts.t_solution_id = :query_id');

    $this->db->bindQueryParameter(':query_id', $query_id);

    $resultRow = $this->db->getSingleResult();
    return $resultRow;
}

private function getQueryComponents($query_id) {
    $this->db->setQuery('SELECT *
                    FROM `components`
                    WHERE query_id = :query_id');

    $this->db->bindQueryParameter(':query_id', $query_id);
```

```
        $resultRows = $this->db->getArrayResult();
        return $resultRows;
    }

    public function getQueryComponentsCount($query_id) {
        $this->db->setQuery('SELECT COUNT(*) AS components_count
                    FROM `components`
                    WHERE query_id = :query_id');

        $this->db->bindQueryParameter(':query_id', $query_id);

        $rowsCount = $this->db->getSingleResult();
        return $rowsCount;
    }

    private function simpleSelectCheck($answerQuery, $queryComponents, &$hints) {
        $operators = [];
        foreach($queryComponents as $queryComponent) {
            array_push($operators, $queryComponent->operator);
        }

        $operatorsStr = strtolower(implode("|(", $operators));
        $answerQueryStatements = preg_split("/(" . $operatorsStr . ")/", strtolower($answerQuery), -1,
PREG_SPLIT_NO_EMPTY);

        for($i = 0; $i < count($queryComponents); $i++) {
            $queryStatement = str_replace("`", "", strtolower($queryComponents[$i]->statement));
            $answerQueryStatement = str_replace("`", "", $answerQueryStatements[$i]);

            if(trim($queryStatement) !== trim($answerQueryStatement)) {
                array_push($hints, $queryComponents[$i]->hint);
            }
        }
    }

    private function joinSelectCheck($answerQuery, $queryComponents, &$hints) {
        $operators = [];
        foreach($queryComponents as $queryComponent) {
            array_push($operators, $queryComponent->operator);
        }
```

```
        $operatorsStr = strtolower(implode(")|(", $operators));
        $answerQueryStatements = preg_split("/(" . $operatorsStr . ")/", strtolower($answerQuery), -1,
PREG_SPLIT_NO_EMPTY);

        for($i = 0; $i < count($queryComponents); $i++) {
            $queryStatement = str_replace("`", "", strtolower($queryComponents[$i]->statement));
            $answerQueryStatement = str_replace("`", "", $answerQueryStatements[$i]);

            if(trim($queryStatement) !== trim($answerQueryStatement)) {
                array_push($hints, $queryComponents[$i]->hint);
            }
        }
    }

    private function createCheck($answerQuery, $queryComponents, &$hints) {
        $operators = [];
        foreach($queryComponents as $queryComponent) {
            array_push($operators, $queryComponent->operator);
        }

        $operatorsStr = strtolower(implode(")|(", $operators));
        $answerQueryStatements = preg_split("/(" . $operatorsStr . ")/", strtolower($answerQuery), -1,
PREG_SPLIT_NO_EMPTY);

        for($i = 0; $i < count($queryComponents); $i++) {
            $queryStatement = str_replace("`", "", strtolower($queryComponents[$i]->statement));
            $answerQueryStatement = str_replace("`", "", $answerQueryStatements[$i]);

            if(trim($queryStatement) !== trim($answerQueryStatement)) {
                array_push($hints, $queryComponents[$i]->hint);
            }
        }
    }

    private function plainTextCheck($answerQuery, $queryComponents, &$hints) {
        $component = $queryComponents[0];
        if(strtolower($answerQuery) != strtolower($component->statement)) {
            array_push($hints, $component->hint);
        }
    }
}
```

User.php.

```
<?php
   class User {
      private $db;

      public function __construct() {
         $this->db = new Database;
      }

      public function getAdmins() {
         $this->db->setQuery('SELECT *
                        FROM `users`
                        WHERE is_admin = :is_admin AND processed = :processed');

         $this->db->bindQueryParameter(':is_admin', 1);
         $this->db->bindQueryParameter(':processed', 1);

         $resultRows = $this->db->getArrayResult();
         return $resultRows;
      }

      public function getStudentsWithGroupOnProcessed($processed) {
         $this->db->setQuery('SELECT u.*, g.number AS group_number
                        FROM `users` AS u
                        LEFT JOIN `groups` AS g ON u.group_id = g.group_id
                        WHERE u.is_admin = :is_admin AND u.processed = :processed');

         $this->db->bindQueryParameter(':is_admin', 0);
         $this->db->bindQueryParameter(':processed', $processed);

         $resultRows = $this->db->getArrayResult();
         return $resultRows;
      }

      public function getUserById($id) {
         $this->db->setQuery('SELECT *
                        FROM `users`
                        WHERE user_id = :user_id');

         $this->db->bindQueryParameter(':user_id', $id);
```

```
    $resultRow = $this->db->getSingleResult();
    return $resultRow;
}

public function login($data) {
    $this->db->setQuery('SELECT *
                FROM `users`
                WHERE username = :username');

    $this->db->bindQueryParameter(':username', $data['username']);

    $resultRow = $this->db->getSingleResult();
    $hashedPassword = $resultRow->password;
    $is_processed = boolval($resultRow->processed);
    if(password_verify($data['password'], $hashedPassword)) {
        if($is_processed) {
            return $resultRow;
        } else {
            return NOT_PROCESSED;
        }
    } else {
        return INCORRECT_CREDENTIALS;
    }
}

public function register($data) {
    $this->db->setQuery('INSERT INTO `users` (username, password, name, surname, group_id, is_admin, processed)
                        VALUES (:username, :password, :name, :surname, :group_id, :is_admin, :processed)');

    $this->db->bindQueryParameter(':username', $data['username']);
    $this->db->bindQueryParameter(':password', $data['password']);
    $this->db->bindQueryParameter(':name', $data['name']);
    $this->db->bindQueryParameter(':surname', $data['surname']);
    $this->db->bindQueryParameter(':group_id', $data['group_id']);
    $this->db->bindQueryParameter(':is_admin', 0);
    $this->db->bindQueryParameter(':processed', 0);

    return $this->db->executeQuery();
}

public function update($data) {
    $this->db->setQuery('UPDATE `users`
```

```
                SET username = :username,
                   name = :name,
                   surname = :surname,
                   group_id = :group_id
                WHERE user_id = :user_id');

    $this->db->bindQueryParameter(':username', $data['username']);
    $this->db->bindQueryParameter(':name', $data['name']);
    $this->db->bindQueryParameter(':surname', $data['surname']);
    $this->db->bindQueryParameter(':group_id', $data['group_id']);
    $this->db->bindQueryParameter(':user_id', $data['id']);

    return $this->db->executeQuery();
}

public function accept($id) {
    $this->db->setQuery('UPDATE `users`
                SET processed = :processed
                WHERE user_id = :user_id');

    $this->db->bindQueryParameter(':processed', 1);
    $this->db->bindQueryParameter(':user_id', $id);

    return $this->db->executeQuery();
}

public function makeAdmin($id) {
    $this->db->setQuery('UPDATE `users`
                SET group_id = :group_id,
                   is_admin = :is_admin
                WHERE user_id = :user_id');

    $this->db->bindQueryParameter(':group_id', NULL);
    $this->db->bindQueryParameter(':is_admin', 1);
    $this->db->bindQueryParameter(':user_id', $id);

    return $this->db->executeQuery();
}

public function delete($id) {
    $this->db->setQuery('DELETE FROM `users`
                WHERE user_id = :user_id');
```

```
        $this->db->bindQueryParameter(':user_id', $id);

        return $this->db->executeQuery();
    }
}
```

Лістинг основних файлів контролерів:

Admin_Pages.php.

```
<?php
    require('LaboratoryWorks.php');
    require('Tasks.php');
    require('Users.php');
    require('Groups.php');
    require('Solutions.php');
    require('DatabaseStructures.php');

    class Admin_Pages extends Controller {
        use LaboratoryWorks;
        use Tasks;
        use Users;
        use Groups;
        use Solutions;
        use DatabaseStructures;

        public function __construct() {
            $this->redirectOnNoAcces();

            $this->laboratoryWorkModel = $this->model('LaboratoryWork');
            $this->taskModel = $this->model('Task');
            $this->userModel = $this->model('User');
            $this->groupModel = $this->model('Group');
            $this->solutionsModel = $this->model('Solution');
            $this->databaseStructureModel = $this->model('DatabaseStructure');
        }

        private function redirectOnNoAcces() {
            if(!checkIsAdminLoggeIn()) {
                exit(header('location:' . URLROOT . '/users_auth/login'));
            }
        }
```

```
    public function database_structure() {
        $titles = $this->databaseStructureModel->getTablesTitles();
        $descriptions = $this->databaseStructureModel->getTablesDescriptions();
        $images = $this->databaseStructureModel->getableStructureImagesSrc();

        $data = [
            'titles' => $titles,
            'descriptions' => $descriptions,
            'images' => $images
        ];

        $this->view('admin_pages/database_structure', $data);
    }
}
```

LaboratoryWorks.php.

```
<?php
  trait LaboratoryWorks {
    private $laboratoryWorkModel;

    public function laboratory_works() {
        $labs = $this->laboratoryWorkModel->getLaboratoryWorks();
        $data = [
            'labs' => $labs
        ];

        $this->view('admin_pages/labs_tasks/laboratory_works', $data);
    }

    public function create_lab() {
        $data = [
            'tiileEN' => '',
            'titleUA' => '',
            'startDate' => '',
            'endDate' => '',
            'maxMark' => '',
            'tiileENError' => '',
            'titleUAError' => '',
            'startDateError' => '',
            'endDateError' => '',
```

```
    'maxMarkError' => ''
];

if($_SERVER['REQUEST_METHOD'] == 'POST') {
    $_POST = filter_input_array(INPUT_POST, FILTER_SANITIZE_STRING);

    $data = [
        'titleEN' => trim($_POST['titleEN']),
        'titleUA' => trim($_POST['titleUA']),
        'startDate' => date("Y-m-d", strtotime(trim($_POST['startDate']))),
        'endDate' => date("Y-m-d", strtotime(trim($_POST['endDate']))),
        'maxMark' => trim((string)$_POST['maxMark']),
        'titleENError' => '',
        'titleUAError' => '',
        'startDateError' => '',
        'endDateError' => '',
        'maxMarkError' => ''
    ];

    $markRule = "/^[0-9.]{1,5}$/";

    if(!$data['titleEN']) {
        $data['titleENError'] = 'Enter the title (EN), please.';
    }

    if(!$data['titleUA']) {
        $data['titleUAError'] = 'Enter the title (UA), please.';
    }

    if($data['startDate'] == '1970-01-01') {
        $data['startDateError'] = 'Enter the start date, please.';
    }

    if($data['endDate'] == '1970-01-01') {
        $data['endDateError'] = 'Enter the end date, please.';
    } elseif($data['startDate'] > $data['endDate']) {
        $data['endDateError'] = 'End date should be lower than start date.';
    }

    if($data['maxMark'] === '') {
        $data['maxMarkError'] = 'Enter the max mark value, please.';
     }elseif(!preg_match($markRule, $data['maxMark'])) {
```

```
            $data['maxMarkError'] = 'Max mark must be an integer or floating point number.';
        }

        if(!$data['titleENError'] && !$data['titleUAError'] &&
            !$data['startDateError'] && !$data['endDateError'] &&
            !$data['maxMarkError']) {
            if($this->laboratoryWorkModel->insert($data)) {
                header('location:' . URLROOT . '/admin_pages/labs_tasks/laboratory_works');
            } else {
                die("Something went wrong...");
            }
        }
    }

    $this->view('admin_pages/labs_tasks/create_lab', $data);
  }

  public function delete_lab($lab_id) {
    if($_SERVER['REQUEST_METHOD'] == 'POST') {
        if($this->laboratoryWorkModel->delete($lab_id)) {
            header('location:' . URLROOT . '/admin_pages/labs_tasks/laboratory_works');
        }
        else {
            die("Something went wrong...");
        }
    }
  }
}
```

Tasks.php.

```
<?php
  trait Tasks {
    private $taskModel;

    public function lab_data($lab_id) {
        $lab = $this->laboratoryWorkModel->getLaboratoryWorkById($lab_id);
        $tasks = $this->taskModel->getTasksByLab($lab_id);
        $data = [
            'lab_id' => $lab_id,
            'lab_title' => $lab->title_en,
            'lab' => $lab,
```

```
    'tasks' => $tasks
];

if($_SERVER['REQUEST_METHOD'] == 'POST') {
    $_POST = filter_input_array(INPUT_POST, FILTER_SANITIZE_STRING);

    $data = [
        'lab_id' => $lab_id,
        'titleEN' => trim($_POST['titleEN']),
        'titleUA' => trim($_POST['titleUA']),
        'startDate' => date("Y-m-d", strtotime(trim($_POST['startDate']))),
        'endDate' => date("Y-m-d", strtotime(trim($_POST['endDate']))),
        'maxMark' => trim((string)$_POST['maxMark']),
        'lab_title' => $lab->title_en,
        'tasks' => $tasks,
        'titleENError' => '',
        'titleUAError' => '',
        'startDateError' => '',
        'endDateError' => '',
        'maxMarkError' => ''
    ];

    $markRule = "/^[0-9.]{1,5}$/";

    if(!$data['titleEN']) {
        $data['titleENError'] = 'Enter the title (EN), please.';
    }

    if(!$data['titleUA']) {
        $data['titleUAError'] = 'Enter the title (UA), please.';
    }

    if($data['startDate'] == '1970-01-01') {
        $data['startDateError'] = 'Enter the start date, please.';
    }

    if($data['endDate'] == '1970-01-01') {
        $data['endDateError'] = 'Enter the end date, please.';
    } elseif($data['startDate'] > $data['endDate']) {
        $data['endDateError'] = 'End date should be lower than start date.';
    }
```

```
        if($data['maxMark'] === '') {
            $data['maxMarkError'] = 'Enter the max mark value, please.';
         }elseif(!preg_match($markRule, $data['maxMark'])) {
            $data['maxMarkError'] = 'Max mark must be an integer or floating point number.';
        }

        if(!$data['titleENError'] && !$data['titleUAError'] &&
            !$data['startDateError'] && !$data['endDateError'] &&
            !$data['maxMarkError']) {
            if($this->laboratoryWorkModel->update(array_slice($data, 0, 6))) {
                header('location:' . URLROOT . '/admin_pages/labs_tasks/laboratory_works');
            } else {
                die("Something went wrong...");
            }
        }
    }

    $this->view('admin_pages/labs_tasks/lab_data', $data);
}

public function create_task($lab_id) {
    $lab = $this->laboratoryWorkModel->getLaboratoryWorkById($lab_id);

    $data = [
        'number' => '',
        'exerciseEN' => '',
        'exerciseUA' => '',
        'difficulty' => '',
        'attempts' => '',
        'lab_id' => $lab_id,
        'lab_title' => $lab->title_en,
        'numberError' => '',
        'exerciseENError' => '',
        'exerciseUAError' => '',
        'difficultyError' => '',
        'attemptsError' => ''
    ];

    if($_SERVER['REQUEST_METHOD'] == 'POST') {
        $_POST = filter_input_array(INPUT_POST, FILTER_SANITIZE_STRING);

        $data = [
```

```
    'number' => trim($_POST['number']),
    'exerciseEN' => trim($_POST['exerciseEN']),
    'exerciseUA' => trim($_POST['exerciseUA']),
    'difficulty' => trim($_POST['difficulty']),
    'attempts' => trim($_POST['attempts']),
    'lab_id' => $lab_id,
    'lab_title' => $lab->title_en,
    'numberError' => '',
    'exerciseENError' => '',
    'exerciseUAError' => '',
    'difficultyError' => '',
    'attemptsError' => ''
];

$numberRule = '/^[0-9.]{1,20}$/';
$difficultyAttemptsRule = '/^[1-9]{1}$/u';

if(!$data['number']) {
    $data['numberError'] = 'Enter the number, please.';
} elseif(!preg_match($numberRule, $data['number'])) {
    $data['numberError'] = 'Number must consist of numbers (integer/floating point).';
}

if(!$data['exerciseEN']) {
    $data['exerciseENError'] = 'Enter the exerise (EN), please.';
}

if(!$data['exerciseUA']) {
    $data['exerciseUAError'] = 'Enter the exerise (UA), please.';
}

if(!$data['difficulty']) {
    $data['difficultyError'] = 'Enter the difficulty, please.';
} elseif(!preg_match($difficultyAttemptsRule, $data['difficulty'])) {
    $data['difficultyError'] = 'Difficulty value must be a not null digit (1-9).';
}

if(!$data['attempts']) {
    $data['attemptsError'] = 'Enter the attempts, please.';
} elseif(!preg_match($difficultyAttemptsRule, $data['attempts'])) {
    $data['attemptsError'] = 'Attempts value must be a not null digit (1-9).';
}
```

```
        if(!$data['numberError'] && !$data['exerciseENError'] &&
            !$data['exerciseUAError'] && !$data['difficultyError'] &&
            !$data['attemptsError']) {
            if($this->taskModel->insert(array_slice($data, 0, 6))) {
                header('location:' . URLROOT . '/admin_pages/labs_tasks/lab_data/' . $lab_id);
            } else {
                die('Something went wrong.');
            }
        }
    }

    $this->view('admin_pages/labs_tasks/create_task', $data);
}

public function update_task($lab_id, $task_id) {
    $task = $this->taskModel->getTaskById($task_id);

    $data = [
        'task_id' => $task_id,
        'task_number' => $task->number,
        'task' => $task,
        'lab_id' => $lab_id
    ];

    if($_SERVER['REQUEST_METHOD'] == 'POST') {
        $_POST = filter_input_array(INPUT_POST, FILTER_SANITIZE_STRING);

        $data = [
            'task_id' => $task_id,
            'number' => trim($_POST['number']),
            'exerciseEN' => trim($_POST['exerciseEN']),
            'exerciseUA' => trim($_POST['exerciseUA']),
            'difficulty' => trim($_POST['difficulty']),
            'attempts' => trim($_POST['attempts']),
            'task_number' => $task->number,
            'lab_id' => $lab_id,
            'numberError' => '',
            'exerciseENError' => '',
            'exerciseUAError' => '',
            'difficultyError' => '',
            'attemptsError' => ''
```

```
        ];

        $numberRule = "/^[0-9.]{1,20}$/";
        $difficultyAttemptsRule = "/^[1-9]{1}$/u";

        if(!$data['number']) {
            $data['numberError'] = 'Enter the number, please.';
        } elseif(!preg_match($numberRule, $data['number'])) {
            $data['numberError'] = 'Number must consist of numbers (integer/floating point).';
        }

        if(!$data['exerciseEN']) {
            $data['exerciseENError'] = 'Enter the exerise (EN), please.';
        }

        if(!$data['exerciseUA']) {
            $data['exerciseUAError'] = 'Enter the exerise (UA), please.';
        }

        if(!$data['difficulty']) {
            $data['difficultyError'] = 'Enter the difficulty, please.';
        } elseif(!preg_match($difficultyAttemptsRule, $data['difficulty'])) {
            $data['difficultyError'] = 'Difficulty value must be a not null digit (1-9).';
        }

        if(!$data['attempts']) {
            $data['attemptsError'] = 'Enter the attempts, please.';
        } elseif(!preg_match($difficultyAttemptsRule, $data['attempts'])) {
            $data['attemptsError'] = 'Attempts value must be a not null digit (1-9).';
        }

        if(!$data['numberError'] && !$data['exerciseENError'] &&
            !$data['exerciseUAError'] && !$data['difficultyError'] &&
            !$data['attemptsError']) {
            if($this->taskModel->update(array_slice($data, 0, 6))) {
                header('location:' . URLROOT . '/admin_pages/labs_tasks/lab_data/' . $lab_id);
            } else {
                die('Something went wrong.');
            }
        }
    }
```

```
        $this->view('admin_pages/labs_tasks/update_task', $data);
    }

    public function delete_task($lab_id, $task_id) {
        if($_SERVER['REQUEST_METHOD'] == 'POST') {
            if($this->taskModel->delete($task_id)) {
                header('location:' . URLROOT . '/admin_pages/labs_tasks/lab_data/' . $lab_id);
            }
            else {
                die("Something went wrong...");
            }
        }
    }
}
```

Users.php.

```
<?php
    trait Users {
        private $userModel;

        public function new_users() {
            $users = $this->userModel->getStudentsWithGroupOnProcessed(0);
            $groups = $this->groupModel->getAllGroups();
            $data = [
                'users' => $users,
                'groups' => $groups
            ];

            $this->view('admin_pages/users/new_users', $data);
        }

        public function current_users() {
            $admins = $this->userModel->getAdmins();
            $students = $this->userModel->getStudentsWithGroupOnProcessed(1);
            $groups = $this->groupModel->getAllGroups();
            $data = [
                'admins' => $admins,
                'students' => $students,
                'groups' => $groups
            ];
```

```
    $this->view('admin_pages/users/current_users', $data);
}

public function update_user($id) {
    $user = $this->userModel->getUserById($id);
    $group = $this->groupModel->getGroupById($user->group_id);
    $groups = $this->groupModel->getAllGroups();

    $data = [
        'id' => $id,
        'user' => $user,
        'group' => $group,
        'groups' => $groups
    ];

    if($_SERVER['REQUEST_METHOD'] == 'POST') {
        $_POST = filter_input_array(INPUT_POST, FILTER_SANITIZE_STRING);
        $group = $this->groupModel->getGroupByNumber(trim($_POST['group']));

        $data = [
            'id' => $id,
            'username' => trim($_POST['username']),
            'name' => trim($_POST['name']),
            'surname' => trim($_POST['surname']),
            'group_id' => $group->group_id,
            'groups' => $groups,
            'usernameError' => '',
            'nameError' => '',
            'surnameError' => '',
        ];

        $usernameRule = "/^[a-zA-Z0-9_@.]{3,32}$/";
        $nameRule = "/^[a-zA-Za-яA-ЯёЁґҐіІїЇєЄ]*$/u";

        if(!$data['username']) {
            $data['usernameError'] = 'Enter the username, please.';
        } elseif(!preg_match($usernameRule, $data['username'])) {
            $data['usernameError'] = 'Username must be at least 3 and no longer than 32 characters,'.
            ' may contain letters, numbers, underscore, at sign and dot.';
        }

        if(!$data['name']) {
```

```
            $data['nameError'] = 'Enter the name, please.';
        } elseif(!preg_match($nameRule, $data['name'])) {
            $data['name'] = 'Name can only contain letters (English/Russian/Ukrainian).';
        }

        if(!$data['surname']) {
            $data['surnameError'] = 'Enter the surname, please.';
        } elseif(!preg_match($nameRule, $data['surname'])) {
            $data['surnameError'] = 'Surname can only contain letters (English/Russian/Ukrainian).';
        }

        if(!$data['nameError'] && !$data['surnameError'] && !$data['usernameError']) {
            if($this->userModel->update(array_slice($data, 0, 5))) {
                header('location:' . URLROOT . '/admin_pages/users/current_users');
            } else {
                die('Something went wrong.');
            }
        }
    }

    $this->view('admin_pages/users/update_user', $data);
}

public function delete_user($id) {
    if($_SERVER['REQUEST_METHOD'] == 'POST') {
        if($this->userModel->delete($id)) {
            header('location:' . URLROOT . '/admin_pages/users/current_users');
        }
        else {
            die("Something went wrong...");
        }
    }
}

public function make_admin($id) {
    if($_SERVER['REQUEST_METHOD'] == 'POST') {
        if(!$_POST['is_admin']) {
            $this->userModel->makeAdmin($id);

            header('location:' . URLROOT . '/admin_pages/users/current_users');
        }
    }
```

```
    }

    public function acceptNewUsers() {
        if($_SERVER['REQUEST_METHOD'] == 'POST') {
            if($_POST['selectedIds'] != '[]') {
                $selectedIds = json_decode($_POST['selectedIds']);
                foreach($selectedIds as $id) {
                    $this->userModel->accept($id);
                }
            }
        }

        header('location:' . URLROOT . '/admin_pages/users/new_users');
    }

    public function declineNewUsers() {
        if($_SERVER['REQUEST_METHOD'] == 'POST') {
            if($_POST['selectedIds'] != '[]') {
                $selectedIds = json_decode($_POST['selectedIds']);
                foreach($selectedIds as $id) {
                    $this->userModel->delete($id);
                }
            }
        }

        header('location:' . URLROOT . '/admin_pages/users/new_users');
    }
}
```

User_Auth.php.

```
<?php
    class Users_Auth extends Controller {
        private $userModel;
        private $groupModel;

        public function __construct() {
            $this->userModel = $this->model('User');
            $this->groupModel = $this->model('Group');
        }

        private function createSession($user) {
```

```
    $_SESSION['user_id'] = $user->user_id;
    $_SESSION['username'] = $user->username;
    $_SESSION['is_admin'] = $user->is_admin;
}

public function login() {
    $data = [
        'username' => '',
        'password' => '',
        'usernameError' => '',
        'passwordError' => ''
    ];

    if($_SERVER['REQUEST_METHOD'] == 'POST') {
        $_POST = filter_input_array(INPUT_POST, FILTER_SANITIZE_STRING);
        $data = [
            'username' => trim($_POST['username']),
            'password' => trim($_POST['password']),
            'usernameError' => '',
            'passwordError' => ''
        ];

        if(!$data['username']) {
            $data['usernameError'] = 'Enter the username, please.';
        }

        if(!$data['password']) {
            $data['passwordError'] = 'Enter the password, please.';
        }

        if(!$data['usernameError'] && !$data['passwordError']) {
            $loggedInUser = $this->userModel->login(array_slice($data, 0, 2));

            if($loggedInUser == NOT_PROCESSED) {
                $data['passwordError'] = 'You need to wait for confirmation of registration from the teacher.';
            }
            if($loggedInUser == INCORRECT_CREDENTIALS) {
                $data['passwordError'] = 'Username or password is incorrect. Please try again.';
            }
            if(!$data['passwordError']) {
                $this->createSession($loggedInUser);
                if(isAdmin()) {
```

```
                header('location:' . URLROOT . '/admin_pages/labs_tasks/laboratory_works');
            } else {
                header('location:' . URLROOT . '/user_pages/main');
            }
        }
    }

    $this->view('users_auth/login', $data);
}

public function logout() {
    unset($_SESSION['user_id']);
    unset($_SESSION['username']);
    unset($_SESSION['is_admin']);
    header('location:' . URLROOT . '/users_auth/login');
}

public function register() {
    $groups = $this->groupModel->getAllGroups();

    $data = [
        'name' => '',
        'surname' => '',
        'group_id' => '',
        'username' => '',
        'password' => '',
        'repassword' => '',
        'groups' => $groups,
        'nameError' => '',
        'surnameError' => '',
        'usernameError' => '',
        'passwordError' => '',
        'repasswordError' => ''
    ];

    if($_SERVER['REQUEST_METHOD'] == 'POST') {
        $_POST = filter_input_array(INPUT_POST, FILTER_SANITIZE_STRING);
        $group = $this->groupModel->getGroupByNumber(trim($_POST['group']));

        $data = [
            'name' => trim($_POST['name']),
```

```
    'surname' => trim($_POST['surname']),
    'group_id' => $group->group_id,
    'username' => trim($_POST['username']),
    'password' => trim($_POST['password']),
    'repassword' => trim($_POST['repassword']),
    'groups' => $groups,
    'nameError' => '',
    'surnameError' => '',
    'usernameError' => '',
    'passwordError' => '',
    'repasswordError' => ''
];

$nameRule = "/^[a-zA-Za-яA-ЯёЁґіІїЇє€]*$/u";
$usernameRule = "/^[a-zA-Z0-9_@.]{3,32}$/";
//$passwordRule = "/^(?=.{8,32})(?=.*[a-z])(?=.*[A-Z])(?=.*[!@#$&*])(?=.*[0-9])$/";
$passwordRule = "/^(?=\S{8,32})(?=\S*[a-z])(?=\S*[A-Z])(?=\S*[0-9])\S*$/";

if(!$data['name']) {
    $data['nameError'] = 'Enter the name, please.';
} elseif(!preg_match($nameRule, $data['name'])) {
    $data['nameError'] = 'Name can only contain letters (English/Russian/Ukrainian).';
}

if(!$data['surname']) {
    $data['surnameError'] = 'Enter the surname, please.';
} elseif(!preg_match($nameRule, $data['surname'])) {
    $data['surnameError'] = 'Surname can only contain letters (English/Russian/Ukrainian).';
}

if(!$data['username']) {
    $data['usernameError'] = 'Enter the username, please.';
} elseif(!preg_match($usernameRule, $data['username'])) {
    $data['usernameError'] = 'Username must be at least 3 and no longer than 32 characters,'.
    ' may contain letters, numbers, underscore, at sign and dot.';
}

if(!$data['password']) {
    $data['passwordError'] = 'Enter the password, please.';
} elseif(!preg_match($passwordRule, $data['password'])) {
    $data['passwordError'] = ' Password must be at least 8 and no longer than 32 characters,' .
    ' contain uppercase and lowercase letters and numbers';
```

```
            }//and special symbols (!, @, #, $, &, *).

            if(!$data['repassword']) {
                $data['repasswordError'] = 'Confirm the password, please.';
            } else {
                if($data['password'] !== $data['repassword']) {
                    $data['repasswordError'] = 'Passwords do not match.';
                }
            }

            if(!$data['nameError'] && !$data['surnameError'] &&
             !$data['usernameError'] && !$data['passwordError'] &&
             !$data['repasswordError']) {
                $data['password'] = password_hash($data['password'], PASSWORD_DEFAULT);
                $data['repassword'] = $data['password'];
                if($this->userModel->register($data)) {
                    header('location:' . URLROOT . '/users_auth/login');
                } else {
                    die('Something went wrong.');
                }
            }
        }

        $this->view('users_auth/register', $data);
    }
}
```

Solutions.php.

```
<?php
    require(APPROOT . '/libraries/Excel/PHPExcel.php');

    trait Solutions {
        private $solutionsModel;

        public function new_solutions() {
            $solutions = $this->solutionsModel->getNewStudentsSolutions();
            $groups = $this->groupModel->getAllGroups();
            $data = [
                'solutions' => $solutions,
                'groups' => $groups
            ];
```

```
    $this->view('admin_pages/solutions/new_solutions', $data);
}

public function new_solution($solution_id) {
    $solution = $this->solutionsModel->getNewStudentsSolutionById($solution_id);
    $data = [
        'id' => $solution_id,
        'solution' => $solution,
        'markError' => ''
    ];

    if($_SERVER['REQUEST_METHOD'] == 'POST') {
        $_POST = filter_input_array(INPUT_POST, FILTER_SANITIZE_STRING);

        $data = [
            'id' => $solution_id,
            'mark' => trim((string)$_POST['mark']),
            'comment' => trim($_POST['comment']),
            'solution' => $solution,
            'markError' => ''
        ];

        $markRule = '/^[0-9.]{1,5}$/';

        if($data['mark'] === '') {
            $data['markError'] = 'Enter the mark value, please.';
        } elseif(!preg_match($markRule, $data['mark'])) {
            $data['markError'] = 'Mark must be a positive integer or floating point number.';
        } elseif(floatval($data['mark']) > $data['solution']->difficulty) {
            $data['markError'] = 'Mark can not be bigger then maximum value.';
        }

        if(!$data['markError']) {
            if($this->solutionsModel->saveConfirm(array_slice($data, 0, 3))) {
                header('location:' . URLROOT . '/admin_pages/solutions/new_solutions');
            } else {
                die("Something went wrong...");
            }
        }
    }
```

```
    $this->view('admin_pages/solutions/new_solution', $data);
}

public function checked_solutions() {
    $solutions = $this->solutionsModel->getCheckedStudentsSolutions();
    $groups = $this->groupModel->getAllGroups();
    $data = [
        'solutions' => $solutions,
        'groups' => $groups
    ];
    $this->view('admin_pages/solutions/checked_solutions', $data);
}

public function correct_solutions() {
    $solutions = $this->solutionsModel->getCorrectSolutions();
    $data = [
        'solutions' => $solutions
    ];

    $this->view('admin_pages/solutions/correct_solutions', $data);
}

public function confirmNewAnswers() {
    if($_SERVER['REQUEST_METHOD'] == 'POST') {
        if($_POST['selectedIds'] != '[]') {
            $selectedIds = json_decode($_POST['selectedIds']);
            foreach($selectedIds as $id) {
                $this->solutionsModel->confirm($id);
            }
        }
    }

    header('location:' . URLROOT . '/admin_pages/solutions/new_solutions');
}

public function disproveNewAnswers() {
    if($_SERVER['REQUEST_METHOD'] == 'POST') {
        if($_POST['selectedIds'] != '[]') {
            $selectedIds = json_decode($_POST['selectedIds']);
            foreach($selectedIds as $id) {
                $this->solutionsModel->disprove($id);
            }
```

```
        }
    }

    header('location:' . URLROOT . '/admin_pages/solutions/new_solutions');
}

public function disproveSolution($id) {
    if($_SERVER['REQUEST_METHOD'] == 'POST') {
        if($this->solutionsModel->delete($id)) {
            header('location:' . URLROOT . '/admin_pages/solutions/new_solutions');
        }
        else {
            die("Something went wrong...");
        }
    }
}

public function excel_export() {
    if($_SERVER['REQUEST_METHOD'] == 'POST') {
        $_POST = filter_input_array(INPUT_POST, FILTER_SANITIZE_STRING);

        $data = [
            'year' => trim($_POST['year'])
        ];

        $labsByYear = $this->laboratoryWorkModel->getLaboratoryWorksByYear($data['year']);
        $groupsWithSolutionByYear = $this->groupModel->getGroupsWithSolutionsByLabYear($data['year']);

        $phpExcel = new PHPExcel;
        $currentSheetIndex = 0;

        foreach($groupsWithSolutionByYear as $group) {
            $newSheet = $phpExcel->createSheet();

            $group_id = $group->group_id;
            $group_number = $group->number;
            $currentSheet = $phpExcel->setActiveSheetIndex($currentSheetIndex)->setTitle($group_number);
            $startRowIndex = 1;
            $startColumnIndex = 0;
            $columnsNames = ['№', 'Initials and Surname of a Student'];
            $currentColumnIndex = $startColumnIndex;
```

```
            foreach($columnsNames as $columnName) {
                $currentSheet->setCellValueByColumnAndRow($currentColumnIndex, $startRowIndex, $columnName);
                $currentSheet->getStyleByColumnAndRow($currentColumnIndex, $startRowIndex)->getAlignment()->setHorizontal(PHPExcel_Style_Alignment::HORIZONTAL_CENTER);
                $currentColumnIndex++;
            }

            $currentSheet->getColumnDimension('A')->setAutoSize(false);
            $currentSheet->getColumnDimension('A')->setWidth('5');
            $currentSheet->getColumnDimension('B')->setAutoSize(true);

            $labsArray = [];
            $labIndex = 0;
            foreach($labsByYear as $lab) {
                $lab_id = $lab->lab_id;
                $lab_title = $lab->title_en;
                $currentSheet->setCellValueByColumnAndRow($currentColumnIndex, $startRowIndex, $lab_title);
                $columnName = $currentSheet->getCellByColumnAndRow($currentColumnIndex, $startRowIndex)->getColumn();
                //$rowName = $currentSheet->getCellByColumnAndRow($currentColumnIndex, $startRowIndex)->getRow();
                $currentSheet->getStyleByColumnAndRow($currentColumnIndex, $startRowIndex)->getAlignment()->setWrapText(true);
                $currentSheet->getColumnDimension($columnName)->setAutoSize(true);

                $labsArray[$labIndex] = ["id" => $lab_id, "title" => $lab_title, "currentColumnIndex" => $currentColumnIndex];
                $labIndex++;
                $currentColumnIndex++;
            }

            $currentSheetIndex++;
        }

        header('Content-Type: application/vnd.ms-excel');
        header('Content-Disposition: attachment; filename="' . $data['year'] . '.xls"');
        header('Cache-Control: max-age=0');
        $excelWriter = PHPExcel_IOFactory::createWriter($phpExcel, 'Excel5');
        $excelWriter->save('php://output');
    }
  }
}
```

User_Pages.php.

```
<?php
    require('LaboratoryWorks_Student.php');
    require('Tasks_Student.php');
    require('Results.php');
    require('DatabaseStructures_Student.php');

    class User_Pages extends Controller {
        use LaboratoryWorks_Student;
        use Tasks_Student;
        use Results;
        use DatabaseStructures_Student;

        public function __construct() {
            $this->redirectOnNoAcces();

            $this->laboratoryWorkModel = $this->model('LaboratoryWork');
            $this->taskModel = $this->model('Task');
            $this->solutionsModel = $this->model('Solution');
            $this->databaseStructureModel = $this->model('DatabaseStructure');
            $this->taskCheckModel = $this->model('TaskCheck');
        }

        private function redirectOnNoAcces() {
            if(!checkIsUserLoggeIn()) {
                exit(header('location:' . URLROOT . '/users_auth/login'));
            }
        }

        public function database_structure() {
            $titles = $this->databaseStructureModel->getTablesTitles();
            $descriptions = $this->databaseStructureModel->getTablesDescriptions();
            $images = $this->databaseStructureModel->getableStructureImagesSrc();

            $data = [
                'titles' => $titles,
                'descriptions' => $descriptions,
                'images' => $images
            ];
```

```
        $this->view('user_pages/database_structure', $data);
    }
}
```

Task_Student.php.

```
<?php
trait Tasks_Student {
    private $taskModel;
    private $solutionsModel;
    private $taskCheckModel;

    public function lab_tasks($lab_id) {
        $lab = $this->laboratoryWorkModel->getLaboratoryWorkById($lab_id);
        $tasks = $this->taskModel->getTasksByLab($lab_id);
        $tasksAttemptsLeft = $this->solutionsModel->getTasksAttemptsLeft($lab_id, $_SESSION['user_id']);
        $tasksBestScores = $this->solutionsModel->getTasksBestScores($lab_id, $_SESSION['user_id']);

        $data = [
            'lab' => $lab,
            'tasks' => $tasks,
            'tasksAttemptsLeft' => $tasksAttemptsLeft,
            'tasksBestScores' => $tasksBestScores
        ];

        $this->view('user_pages/labs_tasks/lab_tasks', $data);
    }

    public function lab_task($lab_id, $task_id) {
        $task = $this->taskModel->getTaskById($task_id);
        $prevNextIds = $this->taskModel->getNextAndPreviousTasksIds($lab_id, $task_id);
        $solutions = $this->solutionsModel->getStudentAttemptSolutions($task_id, $_SESSION['user_id']);
        $solutionsCount = $this->solutionsModel->getStudentAttemptsOnTaskCount($task_id, $_SESSION['user_id']);
        $attemptsLeft = $task->attempts - (int)$solutionsCount->solutions_count;

        $data = [
            'lab_id' => $lab_id,
            'task' => $task,
            'prevTaskId' => $prevNextIds[0]->nptask_id,
            'nextTaskId' => $prevNextIds[1]->nptask_id,
            'solutions' => $solutions,
            'attemptsLeft' => $attemptsLeft,
```

```
            'answer' => "",
             'isAnswerCorrect' => NULL,
             'errorDetail' => ""
        ];

        if($_SERVER['REQUEST_METHOD'] == 'POST') {
            $_POST = filter_input_array(INPUT_POST, FILTER_SANITIZE_STRING);

            $data = [
                'lab_id' => $lab_id,
                'task' => $task,
                'prevTaskId' => $prevNextIds[0]->nptask_id,
                'nextTaskId' => $prevNextIds[1]->nptask_id,
                'solutions' => $solutions,
                'attemptsLeft' => $attemptsLeft,
                'answer' => trim($_POST['answerTextArea']),
                 'isAnswerCorrect' => NULL,
                 'errorDetail' => ""
            ];

            /* Check answer start. */
            $mark = $task->difficulty;
            $comment = "Succes.";
            $correctSolution = $this->solutionsModel->getCorrectSolutionByTaskId($task_id);
            $querySyntaxCheckRes = $this->taskCheckModel->checkQuerySyntax($data['answer']);
            if($querySyntaxCheckRes === true) {
                $querySemanticsCheckRes = $this->taskCheckModel->checkQuerySemantics($data['answer'], $correctSolution-
>t_solution_id);
                if($querySemanticsCheckRes === true) {
                    $data['isAnswerCorrect'] = true;
                } else {
                     $data['isAnswerCorrect'] = false;
                     $data['errorDetail'] = implode(" ", $querySemanticsCheckRes);
                    $componentsCountRes = $this->taskCheckModel->getQueryComponentsCount($correctSolution-
>t_solution_id);
                    $componentsCount = (int)$componentsCountRes->components_count;
                    $mistakesCount = count($querySemanticsCheckRes);
                    $mark -= $mark * ($mistakesCount / $componentsCount);
                    $comment = "Semantics error.";
                }
            } else {
                $data['isAnswerCorrect'] = false;
```

```
                $data['errorDetail'] = $querySyntaxCheckRes[2];
                $mark = 0;
                $comment = "Syntax error.";
            }

            /* Check answer end. */

            $dataToInsert = [
                'answer' => $data['answer'],
                'user_id' => $_SESSION['user_id'],
                'lab_id' => $lab_id,
                'task_id' => $task_id,
                'mark' => $mark,
                'comment' => $comment
            ];

            if($this->solutionsModel->insert($dataToInsert)) {
                header("Refresh:0");
            } else {
                die('Something went wrong.');
            }
        }

        $this->view('user_pages/labs_tasks/lab_task', $data);
    }
}
```

Лістинг основних файлів вигляду:

laboratory_works.php.

```
<?php require APPROOT . '/views/includes/header.php'; ?>
<?php require APPROOT . '/views/includes/common_res.php'; ?>
<?php require APPROOT . '/views/includes/admin_navigation.php'; ?>

<div class="container mt-4">
    <div class="text-center mb-4">
        <h2 class="text-primary"><u>Laboratory Works</u></h2>
    </div>
    <table class="table table-hover">
        <thead class="table-light">
            <tr class="text-center">
                <th class="sortable" scope="col" style="width: 40%">Title</th>
```

```
            <th class="sortable" scope="col" style="width: 22.5%">Start Date</th>
            <th class="sortable" scope="col" style="width: 22.5%">End Date</th>
            <th class="sortable" scope="col" style="width: 15%">Maximum mark</th>
          </tr>
        </thead>
        <tbody>
          <?php foreach($data['labs'] as $lab): ?>
            <tr class="text-center clickable" onclick="window.location='<?= URLROOT
?>/admin_pages/labs_tasks/lab_data/<?= $lab->lab_id ?>'">
              <td class="text-start"><?= $lab->title_en ?></td>
              <td><?= date("Y-m-d", strtotime($lab->start_date)) ?></td>
              <td><?= date("Y-m-d", strtotime($lab->end_date)) ?></td>
              <td><?= $lab->max_mark ?></td>
            </tr>
          <?php endforeach; ?>
        </tbody>
    </table>
    <div class="text-end mt-4">
      <a class="btn btn-primary" href="<?= URLROOT; ?>/admin_pages/labs_tasks/create_lab">Create Laboratory Work</a>
    </div>
</div>

<?php require APPROOT . '/views/includes/footer.php'; ?>
```

new_solutions.php.

```
<?php require APPROOT . '/views/includes/header.php'; ?>
<?php require APPROOT . '/views/includes/common_res.php'; ?>
<?php require APPROOT . '/views/includes/solutions_res.php'; ?>
<?php require APPROOT . '/views/includes/admin_navigation.php'; ?>

<div class="container-fluid mt-4 mb-5">
    <div class="text-center mb-4">
      <h2 class="text-primary"><u>New Solutions</u></h2>
    </div>
    <table class="table table-hover">
      <thead class="table-light">
        <tr class="text-center">
          <th class="sortable" scope="col" style="width: 15%">Student</th>
          <th class="sortable" scope="col" style="width: 10%">Group</th>
          <th class="sortable" scope="col" style="width: 20%">Work</th>
          <th class="sortable" scope="col" style="width: 10%">Task</th>
```

```
            <th class="sortable" scope="col" style="width: 30%">Student Answer</th>
            <th class="sortable" scope="col" style="width: 10%">Prefedined Mark</th>
            <th class="sortable" style="width: 5%"></th>
        </tr>
        <tr class="text-center">
            <th scope="col" style="width: 15%">
                <div>
                    <input class="form-control" id="nameSurnameFilter" type="search" placeholder="Search by
name/surname..."/>
                </div>
            </th>
            <th scope="col" style="width: 10%">
                <div>
                    <select class="form-select" id="groupFilter">
                        <option value=""></option>
                        <?php foreach($data['groups'] as $group): ?>
                            <?php echo "<option value=" . $group->number . ">" . $group->number . "</option>"; ?>;
                        <?php endforeach; ?>
                    </select>
                </div>
            </th>
            <th scope="col" style="width: 20%">
                <div>
                    <input class="form-control" id="labFilter" type="search" placeholder="Search by labwork..."/>
                </div>
            </th>
            <th scope="col" style="width: 10%">
                <div>
                    <input class="form-control" id="taskFilter" type="search" placeholder="Search by task..."/>
                </div>
            </th>
            <th scope="col" style="width: 30%"></th>
            <th scope="col" style="width: 10%"></th>
            <th class="align-middle">
                <div>
                    <input class="form-check-input" id="selectAll" type="checkbox">
                </div>
            </th>
        </tr>
    </thead>
    <tbody id="solutionsTBody">
        <?php foreach($data['solutions'] as $solution): ?>
```

```
            <tr class="text-center" onclick="window.location='<?= URLROOT ?>/admin_pages/solutions/new_solution/<?=
$solution->s_solution_id ?>'">
                <td class="d-none s-id"><?= $solution->s_solution_id ?></td>
                <td class="text-wrap s-namesurname"><?= $solution->name . " " . $solution->surname  ?></td>
                <td class="s-group"><?= $solution->group_number ?></td>
                <td class="text-start text-wrap s-lab"><?= $solution->lab_title ?></td>
                <td class="s-task"><?= $solution->task_number ?></td>
                <td class="text-start text-wrap"><?= $solution->answer ?></td>
                <td><?= $solution->mark ?>/<?= $solution->difficulty ?></td>
                <td class="align-middle" onclick="event.stopPropagation();">
                    <input class="form-check-input" type="checkbox">
                </td>
            </tr>
        <?php endforeach; ?>
    </tbody>
  </table>
  <div class="text-end mt-4">
    <form id="solutionProcessForm" method="POST">
      <input type="hidden" name="selectedIds" id="selectedIds"/>
      <button class="btn btn-primary me-2" type="submit" formaction="<?= URLROOT;
?>/admin_pages/confirmNewAnswers">Confirm selected</button>
      <button class="btn btn-danger" type="submit" formaction="<?= URLROOT;
?>/admin_pages/disproveNewAnswers">Disprove selected</button>
    </form>
  </div>
</div>

<?php require APPROOT . '/views/includes/footer.php'; ?>
```

login.php.

```
<?php require APPROOT . '/views/includes/header.php'; ?>
<?php require APPROOT . '/views/includes/common_res.php'; ?>

<div class="container">
  <div class="row h-100 align-items-center">
    <div class="col-4 offset-4">
      <div class="d-flex justify-content-center">
      <h2>Sign in</h2>
      </div>
      <form action="<?= URLROOT; ?>/users_auth/login" method="POST">
        <div class="form-group">
```

```
            <div class="col-8 offset-2 mb-2">
              <label for="username">Username</label>
              <input type="text" class="form-control <?php if($data['usernameError']) { ?>is-invalid<?php } ?>"
                  id="username" name="username" placeholder="Enter username"
                  value="<?php echo ($_POST['username'] ? trim($_POST['username']) : $data['username']); ?>">
              <div class="invalid-feedback"><?= $data['usernameError']; ?></div>
            </div>
          </div>
          <div class="form-group mb-4">
            <div class="col-8 offset-2">
              <label for="password">Password</label>
              <input type="password" class="form-control <?php if($data['passwordError']) { ?>is-invalid<?php } ?>"
                  id="password" name="password" placeholder="Password"
                  value="<?php echo ($_POST['pasword'] ? trim($_POST['password']) : $data['password']); ?>">
              <div class="invalid-feedback"><?= $data['passwordError']; ?></div>
            </div>
          </div>
          <div class="col-8 offset-2 mb-2">
            <button type="submit" class="btn btn-primary col-12">Sign in</button>
          </div>
          <div class="col-8 offset-2">
            <p class="text-center">First time here?<br/><a href="<?= URLROOT ?>/users_auth/register">Let`s
register!</a></p>
          </div>
        </form>
      </div>
    </div>
</div>

<?php require APPROOT . '/views/includes/footer.php'; ?>
```

login.php.

```
<?php require APPROOT . '/views/includes/header.php'; ?>
<?php require APPROOT . '/views/includes/common_res.php'; ?>
<?php require APPROOT . '/views/includes/tasksolution_res.php'; ?>
<?php require APPROOT . '/views/includes/user_navigation.php'; ?>

<div class="container-fluid">
    <div class="col-8 offset-2">
      <div class="d-block mt-4">
        <div class="d-flex mb-4">
```

```
            <div class="col-2 text-start align-self-center fs-5">
                <?php if($data['prevTaskId'] !== "0"): ?>
                    <a href="<?= URLROOT; ?>/user_pages/labs_tasks/lab_task/<?= $data['task']->lab_id ?>/<?=
$data['prevTaskId'] ?>">Previous</a>
                <?php endif; ?>
            </div>
            <div class="col-8 text-primary align-self-center text-center fs-3"><u>Task №<?= $data['task']->number
?></u></div>
            <div class="col-2 text-end align-self-center fs-5">
                <?php if($data['nextTaskId'] !== "0"): ?>
                    <a href="<?= URLROOT; ?>/user_pages/labs_tasks/lab_task/<?= $data['task']->lab_id ?>/<?=
$data['nextTaskId'] ?>">Next</a>
                <?php endif; ?>
            </div>
        </div>
        <p class="text-justify fs-5 ms-2"><?= $data['task']->exercise_en ?></p>
        <p class="text-justify fs-5 ms-2"><u>Difficulty:</u>
            <?php for($i = 0; $i < $data['task']->difficulty; $i++) : ?>
                <img src="<?php URLROOT; ?>/public/img/star.png" width=20 height=20>
            <?php endfor; ?>
        </p>
        <hr/>
        <p class="text-justify fs-5 ms-2"><u>Attempts left:</u> <?= $data['attemptsLeft'] ?></p>
    </div>
    <?php if(!is_null($data['isAnswerCorrect'])): ?>
        <?php if($data['isAnswerCorrect']): ?>
            <div class="alert alert-primary" id="alertSuccess" role="alert">
                Your answer have been succesfully processed. Please, wait for teacher confirmation.
            </div>
        <?php else: ?>
            <div class="alert alert-danger" id="alertFault" role="alert">
                Your solution does not meet the requirements. Please, revise it.
                <hr/>
                Prompt: <?= $data['errorDetail'] ?>
            </div>
        <?php endif ?>
    <?php endif ?>
    <form id="targetForm" action="<?= URLROOT; ?>/user_pages/labs_tasks/lab_task/<?= $data['task']->lab_id ?>/<?=
$data['task']->task_id ?>"  method="POST">
        <div class="form-group mb-3">
            <label for="answerTextArea" class="text-primary fs-4 mb-1 ms-2">Your Answer</label>
```

```
            <textarea class="form-control" name="answerTextArea" id="answerTextArea" rows='5' placeholder="Enter your answer here..."><?= $data['answer'] ?></textarea>
        </div>
        <div class="d-flex">
            <div class="input-group">
                <div class="input-group-append me-5">
                    <button type="button" class="btn btn-secondary" id="selectAll">SELECT *</button>
                    <button type="button" class="btn btn-secondary" id="select">SELECT</button>
                    <button type="button" class="btn btn-secondary" id="insert">INSERT</button>
                    <button type="button" class="btn btn-secondary" id="update">UPDATE</button>
                    <button type="button" class="btn btn-secondary" id="delete">DELETE</button>
                    <button type="button" class="btn btn-warning ms-4" id="clear">CLEAR</button>
                </div>
            </div>
            <div class="ml-auto text-nowrap">
                <button id="confirmBtn" type="submit" class="btn btn-primary btn-lg me-auto
                <?php if($data['attemptsLeft'] < 1): ?>
                    disabled
                <?php endif; ?>">
                    CONFIRM ANSWER</button>
            </div>
        </div>
    </form>
    <div class="mt-5 mb-5" id="attemptsDiv">
        <div class="text-center mb-4">
            <h2 class="text-primary"><u>Attempts</u></h2>
        </div>
        <table class="table table-hover">
            <thead class="table-light">
                <tr class="text-center">
                    <th scope="col" style="width: 15%">Answer Date</th>
                    <th scope="col" style="width: 35%">Answer</th>
                    <th scope="col" style="width: 10%">Mark</th>
                    <th scope="col" style="width: 15%">Status</th>
                    <th scope="col" style="width: 25%">Comment</th>
                </tr>
            </thead>
            <tbody>
                <?php foreach($data['solutions'] as $solution): ?>
                    <tr class="text-center">
                        <td class="text-wrap"><?= $solution->answer_date ?></td>
                        <td class="text-wrap"><?= $solution->answer ?></td>
```

```
                    <td><?= $solution->mark ?>/<?= $data['task']->difficulty ?></td>
                    <td>
                        <?php if($solution->confirmed): ?>
                            <span class="text-success">Checked</span>
                        <?php else: ?>
                            <span class="text-warning">On verification...</span>
                        <?php endif ?>
                    </td>
                    <td class="text-wrap"><?= $solution->comment ?></td>
                </tr>
            <?php endforeach; ?>
        </tbody>
    </table>
</div>
</div>
</div>

<?php require APPROOT . '/views/includes/footer.php'; ?>
```