МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Навчальна платформа-тренажер для вивчення баз даних»

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

Керівник роботи — **Берест О. Б.**

Студента групи ІН – 61 — **Черняка І.Ю.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2020 р.

ЗАВДАННЯ
до випускної роботи

Студента четвертого курсу, групи ІН-61 спеціальності "Інформатика" денної форми навчання Черняка Івана Юрійовича.

Тема: "**Навчальна платформа-тренажер для вивчення баз даних**"

Затверджена наказом по СумДУ

№_____ від _____2020 р.

Зміст пояснювальної записки: 1) аналітичний огляд методів створення навчальних курсів; 2) постановка завдання й формування завдань дослідження; 3) опис основних методів, тенденцій та інструментів розробки веб-додатків; 4) розробка інформаційного й програмного забезпечення інтелектуальної системи; 5) аналіз результатів розробки та роботи додатку.

Дата видачі завдання "_____" _____2020 р.

Керівник випускної роботи _____ Берест О. Б.

Завдання приняв до виконання _____ Черняк І. Ю.

РЕФЕРАТ

Записка: 91 стор., 26 рис., 1 табл., 2 додатки, 40 джерел.

Об’єкт дослідження — навчальні платформи та тренажери для вивчення SQL.

Мета роботи — розробка тренажеру для вивчення мови SQL у вигляді веб-додатку з використанням сучасних методів та тенденцій веб-розробки.

Методи дослідження — метод розробки веб-додатку з використанням веб-фреймворку.

Результати — розроблено програмне забезпечення навчальної платформи-тренажеру. При цьому використано розподіл функціоналу додатку на серверну та клієнтську частини та враховано тентенції веб-розробки такі як мобільна версія сайту та автоматичне тестування програмного коду. Платформа реалізована у формі веб-додатку, створеного за допомогою мови програмування Java та веб-фреймворку Spark.

SQL, НАВЧАЛЬНА ПЛАТФОРМА, ТРЕНАЖЕР, ВЕБ-РОЗРОБКА, ВЕБ-ДОДАТОК, АВТОМАТИЧНЕ ТЕСТУВАННЯ, АДАПТИВНИЙ ДИЗАЙН

ЗМІСТ

ВСТУП

Сьогодні, з поширенням доступу до мережі Інтернет, із збільшенням кількості Інтернет-ресурсів, із все більшим переходом бізнесу в онлайн сферу та відповідним зростанням кількості даних, одну з найважливіших ролей починає займати галузь обробки даних.

Але зі збільшенням кількості та масштабу Інтернет-ресурсів не зменшилися вимоги до швидкодії та якості. Зараз затримки в роботі сайту можуть суттєво зменшити задоволення користувача від ресурсу та, як наслідок, відмову від його використання.

Найчастіше розробники сайтів або додатків в Інтернеті для зберігання та обробки даних використовують системи керування базами даних (далі СКДБ). Найпопулярнішими з них є реляційні бази даних [1]. Інструментом же доступу до самих даних є мова SQL. Існує багато реляційних систем керування базами даних з різними відмінностями, але всі вони так чи інакше дотримуються стандартів мови SQL. Тому працюючи з однією СКДБ, можна без великих зусиль почати використовувати іншу.

Масове використання баз даних та їх більші об’єми можуть помітити не лише звичайні користувачі, а й працівники IT-сфери, а саме тестувальники, аналітики, менеджери програмного продукту та інші люди, що безпосередньо не пов’язані з написанням коду.

Знання та уміле використання інструментів доступу та обробки даних значно полегшує роботу та підвищує цінність працівника.

Згідно зі статтею [2] співробітник, що впевнено володіє мовою SQL зможе:

- бути більш самостійним і незалежним від колег з професійної точки зору;
- знаходити потрібні дані швидше та у потрібній саме йому формі;
- значно краще розуміти як влаштована схема даних та сам програмний продукт;
- підвищити свою цінність для компанії та як наслідок отримувати більшу заробітну плату.

Саме ж вивчення мови SQL на базовому та середньому рівні не потребує знань мов програмування або наявності технічних знань у галузі комп’ютерних наук.

Одним із способів користувачеві почати вивчати та набувати досвід роботи з SQL можна назвати онлайн-тренажер. Такий підхід може мати ряд переваг:

- доступність для користувача – не потрібно встановлювати додаткове програмне забезпечення, достатньо мати доступ до Інтернету;
- орієнтованість на практику – постійне виконання практичних завдань допоможе користувачам краще опанувати та засвоїти теоретичний матеріал;
- масовість – можливість одночасно залучати велику кількість людей до вивчення матеріалу й здобуття практичних навичок;
- відкритість – сучасні онлайн-платформи часто мають свої форуми та обговорення, де користувачі можуть обмінюватися своїм досвідом;
- інтерактивність – виконання завдань на оцінку, збільшення свого рейтингу серед інших користувачів, система досягнень тощо можуть значно підвищити зацікавленість користувача та збільшити його лояльність до ресурсу.

Отже, метою даної роботи можна вважати розробку онлайн-платформи для вивчення основ роботи з базами даних за допомогою мови SQL, що зможе покращити практичні навички користувачів у написанні запитів та допомогти їм підвищити свій професійний рівень.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Огляд вакансій IT-ринку

Оцінюючи ситуацію на ринку, можна помітити, що зараз майже в кожній вакансії для розробника, а особливо для веб-галузі, потребують знань реляційних баз знань та SQL. Також кількість запитів за ключовим словом «SQL» значно перевищує кількість запитів за іншими популярними мовами програмування [3] (Таблиця 1.1).

Таблиця 1.1 Кількість пошукових результатів на порталі Dou.ua

Пошуковий запит	**Кількість результатів**
SQL	717
JavaScript	715
Python	526
Java	443
C#	338

Усе це говорить про високий попит на фахівців, що можуть не лише писати й супроводжувати програмний код, а вміти працювати із СКДБ.

Саме тому проблема навчання та отримання практичних навичок застосування SQL сьогодні постає вкрай гостро перед працівниками IT-сфери.

1.2 Огляд електронних навчальних платформ

Як зазначалося раніше одним із найефективніших способів навчання для більшості людей може стати навчальна онлайн-платформа.

Це зручний та доступний інструмент, що не потребує від користувача додаткових зусиль для налаштувань власного програмного оточення та встановлення додаткогового програмного забезпечення.

Також за домомогою онлайн-навчання можна стимулювати користувача до самостійного пошуку та опрацювання навчальних матеріалів, що може значно підвищити його продуктивність та самостійність у

вирішенні поставлених задач як то на навчальному курсі, так і в процесі реальної роботи.

Такий підхід було помічено на навчальній платформі Stepik. Це навчальна платформа від російських розробників, що надає можливість користувачам безкоштовно пройти курси від провідних навчальних закладів та IT-компаній. Розглянемо один із найпопулярніших курсів на платформі. Це курс «Java. Базовій курс» [4].

Сам курс складається з тем, що розбиті на короткі теоретичні матеріли у вигляді відео-уроків та практичні завдання (рис. 1.1).

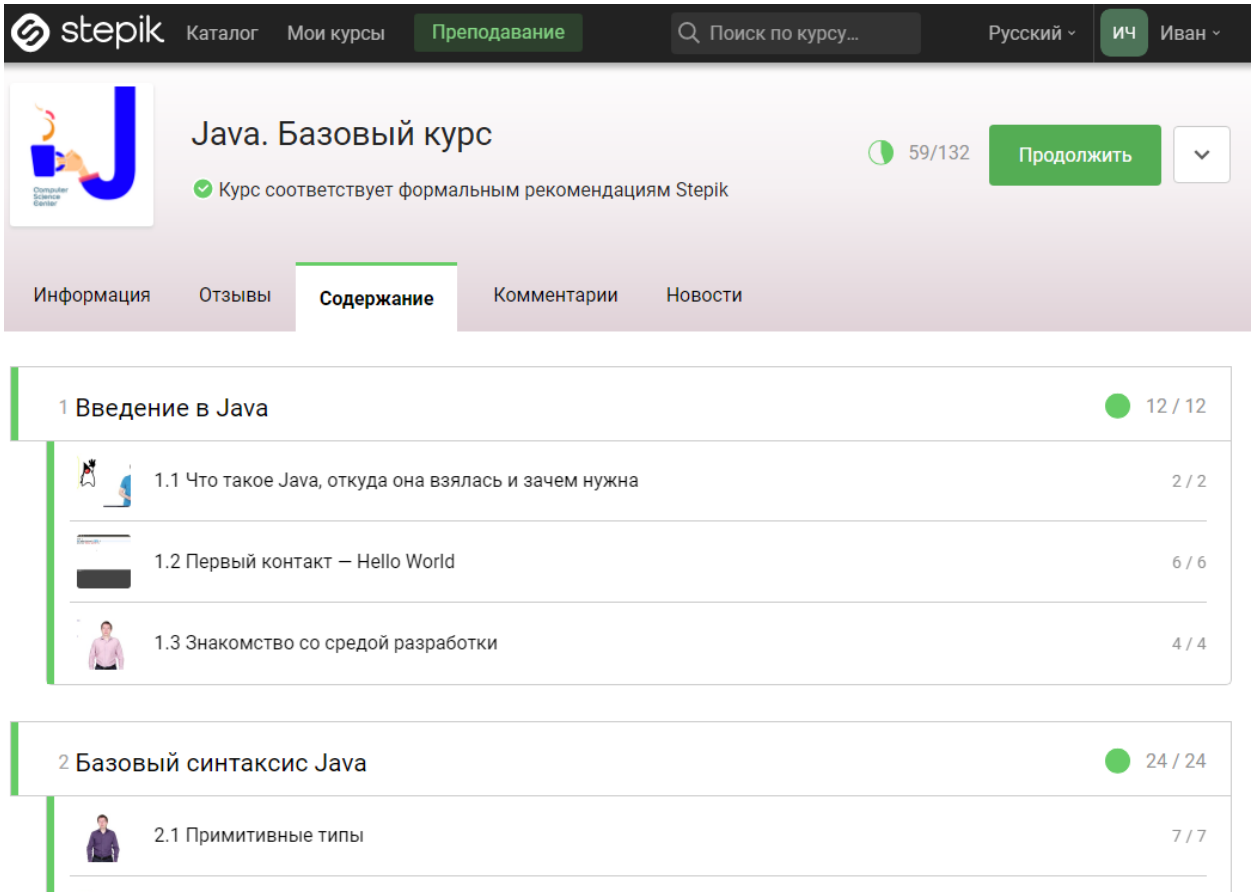

Рисунок 1.1 – Зміст курсу на платформі Stepik

Саме за практичні завдання курс і отримав свою популярність. Для їх вирішення недостатньо просто переглянути відео та повторити код за лектором, вони вимагають від слухача курсу самостійної роботи з матеріалом за темою: пошуку, опрацювання і практичного застосування. Сам курс, по суті, лише задає орієнтири для самостійної роботи з обраною темою.

Але в такого підходу є і свої недоліки:

- досить суттєва розбіжність між матеріалом та прикладами лектора та практичними завданнями, що може відлякати досить суттєву частину аудиторії;
- складність в практичному оцінюванні набутих знань: не завжди можна оцінити якість та цінність самостійно знайденного матеріалу і потрібний погляд зі сторони викладача, але курс не завжди це надає.

Розглянемо приклад іншого підходу до складання навчального курсу.

На українській навчальній платформі Prometheus також є досить популярний курс з основ програмування на мові Java «JAVA101. Основи програмування на Java» [5].

Тут також є можливість проходити тести, виконувати практичні завдання з автоматичною перевіркою та надсилати відповіді на особисту перевірку викладачеві (рис. 1.2).

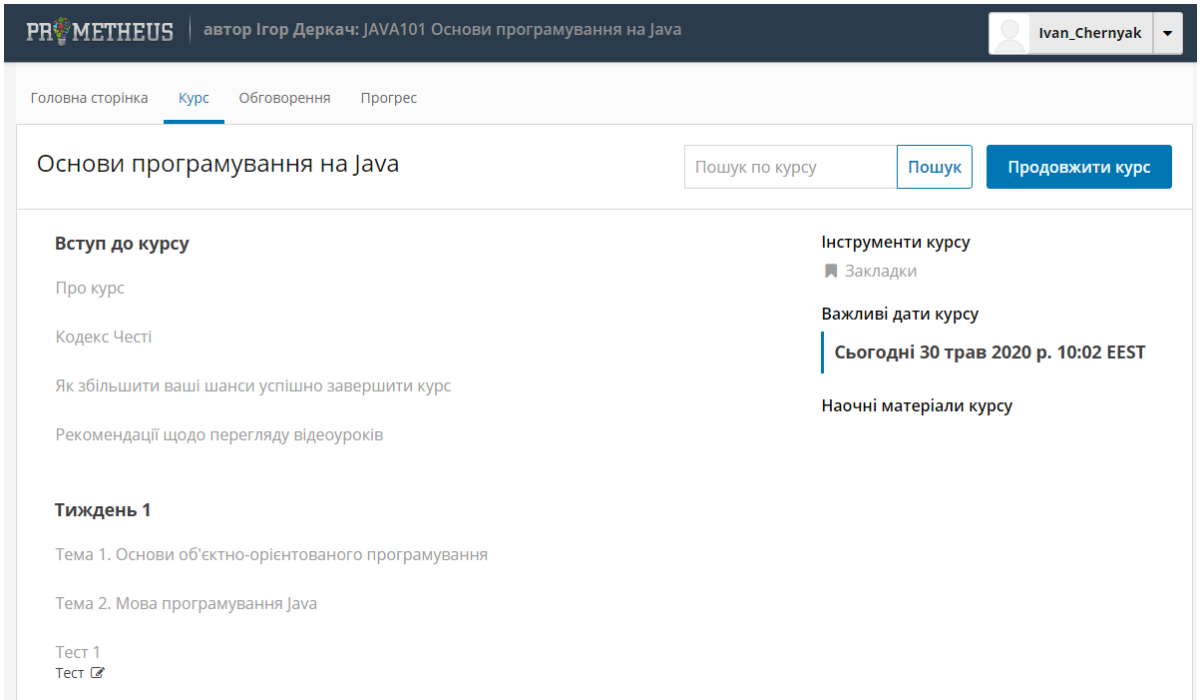

Рисунок 1.2 – Зміст курсу на платформі Prometheus

Але, на відміну від курсу на Stepik, даний курс надає набагато більше теоретичних матеріалів у вигляді відео-лекцій. Вони займають значно більшу

частину курсу. Практичні завдання також містять багато матеріалу, який потрібно самостійно опрацьовувати, але часто він не відповідає навіть темі заняття. Наприклад, при розгляді теми про основи мови, де користувача лише починали знайомити з типами даних та умовними операціями, у якості практичних завдань було запропоновано реалізувати алгоритми сортування. З одного боку – це поштовх до самостійного опрацювання матеріалу в суміжній галузі, а з іншого – це відхід від поданого у теоретичній частині матеріалу.

Беручи до уваги приклади цих курсів, потрібно комбінувати підходи та знаходити баланс між завданнями таким чином, щоб користувач міг як повторити приклади з теоретичної частини, так і почати самойстійно працювати над завданнями.

1.3 Огляд онлайн-тренажарів для вивчення SQL

Розглянемо приклади реалізії застосунків, націлених на вивчення саме SQL.

За запитом «SQL tutorial» у пошуковій системі Google першим ми отримаємо сайт W3Schools [6].

Навчальний курс з SQL тут також поділений на категорії та надає користувачеві як короткі інформаційні відомості з теми, так і практичні приклади та завдання для самостійної перевірки. Але практичні завдання тут обмежені звичайним доповненням запиту (рис. 1.3), що надає користувачу лише перші уявлення про синтаксис мови та її конструкцій.

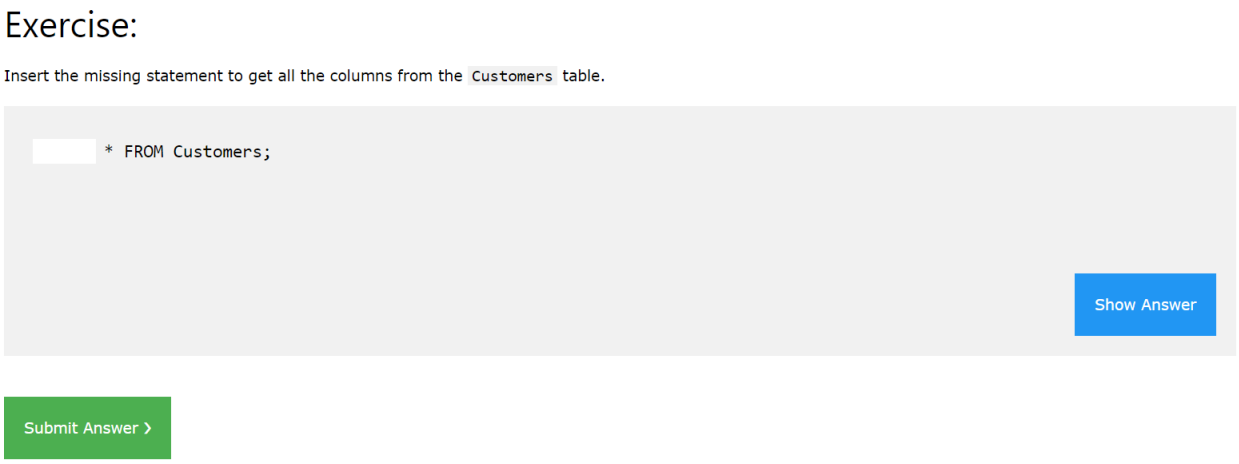

Рисунок 1.3 – Приклад завдання на сайті W3Schools

Оскільки з кожним роком зростає кількість користувачів смартфонів, то також розглянемо які варіанти нам пропонує ринок мобільних додатків для вивчення SQL.

У магазині додатків для Android зробимо пошук за ключовим словом «SQL» та розглянемо найпопулярніші додатки за цим запитом.

Найпершим серед результатів нам видається додаток «Learn SQL» від компанії SoloLearn [7].

У цьому додатку завдання розбиті за категоріями, для кожної категорії є короткий теоретичний опис та самі практичні завдання (рис. 1.4).

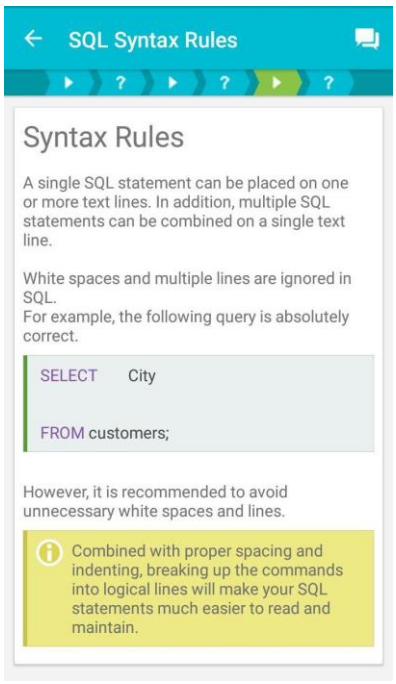

Рисунок 1.4 – Приклад завдання в додатку Learn SQL

Завдання поділяються за типами:

- вибрати правильний варіант відповіді;
- вставити пропущений оператор, щоб запит виконався правильно;
- написати власний запит до бази даних.

Також додаток взаємодіє з користувачем, показуючи підказки та повідомлення (рис. 1.5), а також має систему досягнень та глобальний рейтинг користувачів.

Рисунок 1.5 – Приклад сповіщення в додатку Learn SQL

Розглянувши дані ресурси, можемо зробити висновок, що практичний тренажер повинен мати різні типи завдань, давати короткі та доцільні для практичного використання теоретичні відомості та надавати елементи інтерактивності для користувача.

1.4 Огляд методів веб-розробки

Оскільки планується реалізація онлайн-платформи, то важливо знати і розуміти що таке веб-розробка, веб-сайт та веб-додаток, основні технології та тенденції в розробці.

Веб-додаток – це програма, що використовує веб-браузер та веб-технології для виконання задач через мережу Інтренет.

Веб-сайт у більшості випадків показує статичні або динамічні дані, які переважно надсилаються з сервера користувачеві, тоді як веб-додаток динамічно опрацьовує дані з повною двосторонньою взаємодією: від користувача до сервера та від сервера до користувача [8].

Для веб-додатку потрібен веб-сервер для управління запитами від клієнта, сервер додатків для опрацювання запитів клієнта, а іноді і база даних для зберігання інформації.

Ось як виглядає типовий потік взаємодії користувача із веб-додатком [9]:

1. Користувач відсилає запит на веб-сервер через веб-браузер або інтерфейс програми.
2. Веб-сервер пересилає цей запит на відповідний сервер веб-додатків.
3. Сервер веб-додатків виконує завдання - наприклад, запит до бази даних або обробка даних. Потім він генерує результати своєї роботи.
4. Сервер веб-додатків надсилає результати на веб-сервер із запитуваною інформацією або обробленими даними
5. Веб-сервер надсилає відповідь назад клієнту.

Розглянемо переваги веб-додатку:

- веб-додатки працюють на багатьох платформах, незалежно від ОС або пристрою, головне щоб кінцевий користувач мав браузер;
- вони знижують витрати як для бізнесу, так і для кінцевого споживача, оскільки для бізнесу потрібна менша підтримка та обслуговування, а для до комп'ютера кінцевого користувача - менші вимоги;
- усі користувачі отримують доступ до однієї версії додатку;

1.5 Огляд технологій веб-розробки

Сучасну веб-розробку вже важко уявити без використання декількох фреймворків або так званих стеків технологій – наборів із мов

програмування, бібліотек, утиліт та самих фреймворків, що дозволяють уніфіковано та швидко проводити процес розробки, підвищуючи якість кінцевого продукту.

Сама структура веб-додатку за [10, 11] поділяється на клієтську та серверну частини.

1.5.1 Клієнтська частина веб-додатку

Клієнтська частина додатку – це частина, яку видно користувачам і являє собою «обличчя» або «фасад» додатку. З цієї причини її зазвичай називають «front-end». Технології, пов'язані з цією частиною розробки, не численні:

- мова гіпертекстової розмітки (HTML) управляє структурою інформації, яка відображається користувачу в веб-браузері;
- каскадні таблиці стилів (CSS) визначають стиль відображуваних даних, керуючи такими параметрами як шрифти, розмір і колір тексту та елементів розмітки тощо.
- мова JavaScript управляє інтерактивними веб-функціями програми.

Усі ці технології є потужними інструментами в руках кваліфікованих фахівців. Однак їх поєднання виводить розробників веб-додатків на якісно новий рівень можливостей.

Також використання фреймворків є поширеною практикою, що полегшує процес розробки веб-додатків. Популярними прикладами фреймворків і бібліотек для клієтської частини є:

- Angular – фреймворк, розроблений компанією Google для проектування динамічної структури даних для веб-додатків.
- React – бібліотека від Facebook Inc., орієнтована на проектування інтерфейсів односторінкових додатків, що означає що додаток може показувати різні дані на одній і тій же сторінці без її перезавантаження.
- Vue.js – легкий JavaScript-фреймворк для створення адаптивних користувацьких інтерфейсів для односторінкових веб-додатків, який

швидко завойовує популярність серед розробників з моменту його появи в 2014 році.

Кожен з перерахованих вище інструментів має свій набір переваг і недоліків, спільноту користувачів, а також ряд найбільш ефективних сценаріїв реалізації.

1.5.2 Серверна частина веб-додатку

Серверна сторона веб-додатку або «back-end» прихована від користувачів, але саме вона керує тим, що користувач бачить у веб-браузері.

Компоненти серверної частини включають в себе:

- мову програмування для написання коду веб-додатку. Як і для клієнтської частини, існує кілька мов і численні фреймворки для спрощення і поліпшення процесу кодування. Серед найбільш популярних інструментів для бекенд-програмування можна назвати такі мови як Python, PHP, Ruby, і Java, а також їх фреймворки: Django, Laravel, Ruby on Rails та Spring відповідно.
- базу даних для зберігання даних програми. Вони можуть бути реляційними або нереляційними, у залежності від моделі управління даними. Реляційні бази даних використовують в своїй роботі структуровану мову запитів (SQL), в той час як нереляційні (Non-SQL) бази даних використовують інші моделі для зберігання та вилучення даних. Відповідно до ресурсу DB-Engines, в першу п'ятірку[1]. входять бази даних: Oracle, MySQL, Microsoft SQL Server, PostgreSQL і MongoDB. Відзначимо, що всі вони, крім останньої, є реляційними.
- сервер для обробки запитів, що надходять від клієнтської сторони, тобто від користувачів додатку. Тут вибір досить обмежений: майже всі веб-додатки розробляються з використанням або Nginx, або Apache в якості сервера додатків.

1.6 Огляд тенденцій у веб-розробці

Згідно з [12-14] можемо виділити наступні тенденції при розробці веб-додатків.

1.6.1 Мобільна версія сайту та адаптивний дизайн

Говорячи про тенденції розробки сайтів, неможливо уникнути пункту про адаптивність дизайну та мобільну версію. Сама концепція зародилася ще 5-6 років тому, коли мобільні пристрої почали активно виходити на ринок. Зараз, коли кожен другий пошуковий запит робить користувач мобільного пристрою, ігнорувати таку вимогу до веб-ресурсу просто неможливо.

Це означає, що тепер командам, які створювали веб-додатки, необхідно приділяти більше уваги мобільним версіям сайтів, коли мова йде про дизайн інтерфейсу, а розробники та дизайнери повинні тепер докладати додаткові зусилля для розробки своїх продуктів, щоб зробити їх зручними для користувачів у двох форматах – десктопному та мобільному.

Одним із способів вирішення цієї проблеми є розробка адаптивного веб-дизайну, що використовує один і той же код, який може автоматично змінити візуалізацію, залежно типу пристрою, з якого переглядають сайт.

1.6.2 Онлайн-підтримка та чат-боти

Задля задоволення потреб цілодобової підтримки на сучасних сайтах ми все частіше можемо помітити використання операційних чат-ботів. Сьогодні вони стають усе більш самостійними та підлаштовуються під потреби та поведінку конкретного користувача

Великі B2C-компанії вже використовують цю технологію, щоб обслуговувати своїх клієнтів – офіційні чати сторінок у Facebook, WhatsApp та Skype – хороші приклади.

Бот також може бути інтегрований у звичайний веб-сайт, професійне або побутове обладнання та будь-який додаток, підключений до Інтернету.

Це означає, що 24/7 операційні чат-боти зможуть замінити менеджерів із підтримки та заощадити витрати для багатьох компаній.

Найважливішими ж перевагами ботів є:

- потенціал вирішення проблем цілодобово;
- досвід спілкування, схожий на пілкування з людиною;
- глибока аналітика поведінки користувачів.

1.6.3 Автоматизоване тестування

Більшість нових веб-технологій спрямовані на те, щоб зробити процес розробки дешевшим і надати користувачам найкращий досвід використання ресурсу. Автоматизація етапів розробки – це інструмент для досягнення першої мети.

Одним із головних інструментів у цій сфері є автоматизація тестів. Вона надає переваги у вигляді кращого покриття коду тестами, прозорості програмного продукту та виявленні технічних проблем на ранніх етапах розробки.

Цей підхід також допомагає команді розробників зібрати тестові приклади та вивчити їх, зменшити витрати на розробку та загалом скоротити час циклу тестування.

Іншими словами, за допомогою автоматизації тестування ми можемо отримати кращий кінцевий продукт за менші гроші.

1.7 Постановка задачі

Розробити тренажер для SQL у вигляді веб-додатку, що дозволяє користувачеві:

- зареєструватися та авторизуватися на сайті;
- переглядати та виконувати завдання, що стосуються практичного використання мови SQL;
- переглядати результати виконання завдань;
- переглядати статистику виконаних завдань.

Для досягнення мети можна виділити наступні задачі:

- провести проектування інформаційної системи;

- провести аналіз та побудову моделі програмного продукту;
- реалізувати програмний продукт;
- провести тестування програмного продукту.

Результатом роботи має стати веб-додаток, що працює у всіх сучасних браузерах, має адаптивний дизайн та мобільну версію та є зрозумілим у використанні.

2 ВИБІР МЕТОДІВ РІШЕННЯ ЗАДАЧІ

2.1 Вибір апаратно-програмного інструментарію

Як серверну мову програмування для додатку було обрано Java [15].

Це популярна та проста в засвоєнні мова програмування, що має простий та знайомий синтаксис, що заснований на синтаксисі C++, та буде зрозумілий багатьом розробникам [16].

Це об’єктно-орієнтована мова програмування, що дозволяє у повному обсязі використовувати усі переваги цієї парадигми такі як інкапсуляція, наслідування та поліморфізм.

Java також приділяє чимало уваги продуктивності програми та звільняє розробників від необхідності самостійно контролювати ресурси пам’яті, що надає більше можливостей та часу для реалізації логіки програми.

Java має одну з найбільших спільнот розробників [17], що надає підтримку в разі виникнення будь-якого питання під час роботи з нею.

Сама ж мова дозволяє легко та ефективно реалізовувати логіку будь-якої складності завдяки великій кількості вбудованих інструментів самої мови (колекції, робота з датою та часом, інструменти доступу до бази даних тощо) та сторонніх бібліотек та фреймворків. Самі ж бібліотеки дозволяють заощадити час на розробку, мають відкритий вихідний код, розроблюються та підтримуються досвідченими Java-розробниками.

Усе це робить Java гарним вибором у якості основної мови програмування для реалізації веб-додатку.

У якості веб-фреймворку для Java було обрано Spark Framework [18].

Spark – це простий і виразний веб-фреймворк, створений для швидкої розробки. Намір Spark полягає в тому, щоб надати альтернативу розробникам Java, які хочуть розробляти свої веб-додатки максимально виразно і з мінімальними затратами.

До переваг даного фреймворку можна віднести:

- вбудований веб сервер. Standalone-додаток під управлінням Spark може працювати на веб-сервері Jetty [19], що робить розгортання додатку більш простим;
- простота налаштування. Spark надає широку можливість для налаштувань самого веб-серверу, що може задовольнити конкретні потреби проекту без особливих зусиль.

Також Spark підтримує роботу з багатьма двигунами шаблонів, що дозволяють більш ефективно форматувати динамічне наповнення HTML-документа.

Двигуном шаблонів для проекту було обрано Thymeleaf [20].

Thymeleaf – це серверний двигун шаблонів для мови Java. Він коректно відображається у всіх сучасних браузерах, має зрозумілий та простий синтаксис та досить широкий набір інструментів, таких як змінні, цикли та умовні оператори.

Для системи керування базою даних оберемо PostgreSQL [21].

Це система для управління реляційними базами даних з відкритим вихідним кодом. Для доступу до даних PostgreSQL використовує SQL.

Це досить проста у використанні та підтримці база даних, що може бути запущена на сервері з досить обмеженими ресурсами.

Також дана СКДБ виділяється продуктивністю за рахунок паралельних запитів, великої кількості типів індексів та розширеної оптимізації запитів [22, 23].

PostgreSQL надає своїм користувачам як загальноприйняті інструменти роботи з SQL, так і специфічні особливості, серед головних з яких є:

- віконні та аналітичні функції (оператор OVER);
- рекурсивний SQL (оператор WITH RECURSIVE);
- розширений функціонал для роботи з JSON.

Робота з JSON – це також великий плюс при виборі бази даних, адже використовуючи плоскі таблиці, можна обробити та агрегувати дані у вигляді

JSON-рядка та подати його як результат запиту, що значно полегшує підготовку даних для інтерфейсу користувача та фреймворків.

Не в останню чергу саме через це було обрано засіб для візуалізації статистичних даних користувача – бібліотеку D3.js[24].

Це JavaScript-бібліотека для відображення даних у клієнтському інтерфейсі за допомогою HTML, SVG і CSS, що поєднує потужні компоненти візуалізації і керований підхід до маніпулювання даними.

У якості інструменту для реалізації адаптивного мобільного дизайну було обрано Bootstrap [25].

Це колекція CSS- і JavaScript-шаблонів для розробки інтерфейсних компонентів. Її основна мета – підвищення адаптивності й націленість на мобільні пристрої у веб-розробці.

Для розробки автотестів було обрано фреймворк JUnit 5 [26], бібліотеку JaCoCo [27] та бібліотеку HttpClient [28].

JUnit – це простий фреймворк для написання тестів, що використовує анотації для визначення методів, які проводять тест. JUnit – це проект з відкритим вихідним кодом, розміщений на Github.

JaCoCo – це безкоштовна бібліотека для визначення кількості покритого тестами коду для Java.

HttpClient – це клієнтська транспортна бібліотека HTTP від Apache Software Foundation. Мета HttpClient – передавати і отримувати HTTP-повідомлення.

2.2 Вибір середовища розробки

Для написання та відлагодження програмного коду найкраще використати інтегроване середовище програмування (Integrated Development Environment, далі IDE).

IDE – це повноцінна та складна система, що підтримує усі стадії розробки програмного продукту: від написання коду до розгортання додатку на сервері.

До переваг використання IDE можна віднести:

- мінімальні витрати часу та зусиль, оскільки вся концепція IDE – це зробити простішим та швидшим процес розробки;
- наявність інструментів управління кодом для автоматизації багатьох речей;
- наявність інструментів відлагодження для повного тестування розробленого додатку.

Розглянемо найпопулярніші IDE для написання коду на мові програмування Java. Згідно з [29] це Eclipse, IntelliJ Idea та NetBeans.

NetBeans

Це безкоштовне інтегроване середовище розробки з відкритим кодом, що підтримується Apache Software Foundation. Дозволяє розробляти веб-додатки, застосунки для персональних комп’ютерів, мобільні додатки та ін.

Він поєднується з хорошою архітектурою та вбудованими інструментами, що задовольняють потреби проекту від формулювання вимог до розгортання. У нього є активна спільнота користувачів та розробників у всьому світі. NetBeans пропонує плавне та швидке редагування коду.

Особливості:

- NetBeans працює на ОС Windows, Mac OS, Linux та Solaris.
- інструмент рефакторингу NetBeans дозволяє програмісту реструктурувати код, не порушуючи його;
- NetBeans також виконує аналіз вихідного коду і надає широкий набір підказок щодо покращення коду або швидкого його виправлення;
- він має гарну вбудовану підтримку для Maven and Ant та плагін для Gradle;
- NetBeans пропонує хорошу міжплатформну та багатомовну підтримку;
- він також поставляється із інструментом статичного аналізу та перетворювачами коду.

Eclipse

Це повнофункціональне потужне середовище розробки з відкритим кодом, що широко використовується для розробки програм Java.

Eclipse оснащений базовим робочим простором та розширюваною системою плагінів, за допомогою якої можна налаштувати середовище під конкретного користувача.

Особливості:

- Eclipse є платформою, яка працює на Linux, Mac OS та Windows;
- редагування, перегляд, рефакторинг та налагодження: Eclipse надає всі ці функції та полегшує програмістам розробку програм;
- Eclipse підтримує налагодження як локально, так і віддалено, припускаючи, що ви використовуєте JVM, яка підтримує віддалене налагодження;
- дозволяє інтегруватися з сервером Apache Maven та системою керуванням версіями Git.

IntelliJ IDEA

Це IDE для розробки програмних додатків за допомогою Java. IntelliJ IDEA була розроблена компанією JetBrains. Дане середовище доступне як у безкоштовній, так і в комерційній, версіях. Версії відрізняються наявними інструментами, але основний функціонал доступний в обох версіях.

IntelliJ IDEA дає пропозиції щодо завершення коду, аналізу коду та надійних інструментів рефакторингу.

Особливості:

- розумне завершення: в ньому подано список найбільш релевантних символів, застосовних до поточного контексту. Він постійно пропонує останні використовувані класи, методи, змінні тощо. Таким чином написання коду відбувається швидше;
- аналіз потоку даних: IntelliJ має можливість аналізувати потік даних та відгадувати можливий символ під час виконання;

- IntelliJ пропонує ретельний та ефективний рефакторинг, оскільки він знає все про використання символів;
- IntelliJ IDEA поставляється з великою кількістю вбудованих інструментів, таких як системи контролю версій, декомпілятор, консоль база даних SQL тощо;
- є потужний компілятор, який здатний виявляти дублікати, «запахи» коду тощо.

Intellij IDEA також має безкоштовну учнівську ліцензію для студентів, що беззаперечно є великим плюсом даного продукту.

Для роботи над проектом було обрано Intellij IDEA через більшу кількість переваг, пов’язаних із авто-доповненням та аналізом коду.

3 ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Проектування інформаційної системи

Коли розпочинається робота над проектом, то розробники прагнуть якнайшвидше приступити до написання самого програмного коду. І це нормально, адже це їхня робота. Але дуже часто такий підхід до реалізації програмного продукту може призвести до заплутаного програмного коду та складної архітектури додатку, реалізації непотрібного функціоналу, прострочених термінів сдачі роботи та, як найгірший варіант, до припинення самого проекту.

Тому одним із важливих етапів розробки програмного продукту стає фаза планування та проектування.

Відповідно до статті [30] на цьому етапі можна чітко сформувати вимоги до програмного продукту, продумати архітектуру та програмні модулі для реалізації цих вимог. Також завчасне планування може допомогти виявити слабкі місця додатку, його потенційні проблеми продуктивності та розширення задовго до їх фактичної появи. Це може суттєво зекономити ресурси на подальшій розробці програми.

Залежно від складності поставленої задачі та наявних ресурсів можна використати різні засоби планування: від простої схеми на дошці чи аркуші паперу до оформленої згідно міжнародних стандартів документації.

Також важливо не лише спланувати й розуміти що повинен роюити додаток, а й у доступній, наочній та зрозумілій формі передати цю інформацію тим, хто працює разом з вами над проектам, та тим, хто у подальшому буде супроводжувати вже розроблений програминий продукт.

Одним же із накращих способів донести цю інформацію – це використати графічні інструменти моделювання.

Розглянемо найголовніші з них.

3.1.1 Модель ERD

Для проектування основних сутностей додатку використаємо Entity relation діаграму (EDR). ЇЇ використовують для проектування даних, що будуть зберігатися в базі даних, та їх зв'язків між собою. Дана діаграма проектується однією з перших, показує якими типами даних оперує система, які обмеження цих даних та задає напрями для подальшого проектування.

Компонентами ER-діаграми є:

- сутності – предмети реального чи уявного світу, дані про які необхідно зберігати та обробляти;
- атрибути – властивість або характеристика сутності;
- зв'язки – показують як сутності пов'язані одна з одною та як вони взаємодіють.

Дану діаграму можна використати для проектування як нової бази даних, так і для опису та подальшої підтримки вже існуючої.

Згідно з джерелами [31, 32] серед переваг використання даної моделі можна виділити:

- незалежність від системи керування базами даних;
- візуальне представлення структури бази даних;
- легкість у використанні;
- високу сумісність із реляціними таблицями – готову діаграму можно легко конвертувати.

До недоліків ERD можна віднести:

- відсутність єдиного стандарту нотації;
- складність у зображенні перетворень даних.

3.1.2 Модель IDEF0

Також однією із найпростіших для розуміння та реалізації є графічна нотація IDEF0 [33]**.**

IDEF0 – графічна нотація, яка використовується для виявлення структури і функцій бізнес-процесу, а також показу потоків інформації та

матеріальних ресурсів, які пов'язують ці функції.. IDEF0 є частиною методології структурного аналізу та проектування (Structured Analysis and Design).

До особливостей нотації IDEF0 можна віднести:

- використання контекстної діаграми;
- підтримка декомпозиції;
- виділення 4 типів потоків – Вхід (Input), Управління (Control), Вихід (Output) та Механізм (Mechanism).

3.1.3 Модель UML

UML – це стандартизована мова моделювання, що складається з набору діаграм, розроблених для ідентифікації, візуалізації, конструювання та документування програмних систем [34]. UML представляє собою набір найкращих рішень та практик, що довели свою ефективність у вирішенні проблем моделювання великих і складних систем.

Також використання UML є важливою частиною розробки об’єктно-орієнтованого програмного забезпечення. Використання даного стандарту дозволяє проектним командам легше розуміти один одного, вести комунікацію та працювати ефективніше.

До переваг використання UML при проектуванні можна віднести [35]:

- розповсюдженість. Більшість розробників принаймні знайомі з основними діаграмами та концептами UML і це може значно полегшити комунікацію в команді;
- незалежність від мови програмування. UML надає інструменти, що допоможуть показати функціонал програми, незалежно від того яка мова програмування буде використанна;
- широкі можливості для опису. UML надає велику кількість діаграм, що допоможуть описати систему з різних точок зору та підходів.

Розглянемо безпосередньо саму реалізацію пректування для нашої системи.

3.1.4 Реалізація ERD

Щоб ефективно реалізувати діаграму ER, варто дотримуватися наступних принципів:

1. виділити сутності системи;
2. віділити з’язки між сутностями;
3. описати взаємозв’язки;
4. додати атрибути до сутностей.

Оскільки додаток буде орієнтований на користувачів, то найпершою сутністю можемо віділити саме Користувача. Також система буде пропонувати користувачеві виконати Завдання та переглянути Відповідь. У самих же завдань та користувачів можна виділити Категорії.

Очевидно, що питання можуть мати лише одну категорію й не можуть бути без неї створені. Можемо виділити сильний зв’язок один до багатьох. У той же час категорія користувача та категорія питання можуть бути створені незалежно від інших сутностей.

А от відповіть на питання уже буде залежати як від користувача, так і від завдання. Знову можемо виділити сильний з’язок.

Визначимо атрибути кожної сутності.

- Користувач:
  - порядковий номер;
  - ім’я;
  - логін;
  - пароль;
  - дата реєстрації;
  - рейтинг користувача;

- Категорія користувача:
  - порядковий номер;
  - назва;
  - мінамальний рейтинг;
  - максимальний рейтинг;
  - логотип;
- Категорія питання:
  - порядковий номер;
  - назва;
  - опис;
  - логотип;
- Завдання:
  - порядковий номер;
  - категорія;
  - питання;
  - відповідь;
  - рейтинг питання;
- Відповідь:
  - порядковий номер;
  - порядковий номер користувача;
  - порядковий номер відповіді;
  - дата відповіді.

Тепер можемо завершити створення діаграми, результат можемо побичити на рисунку 3.1.

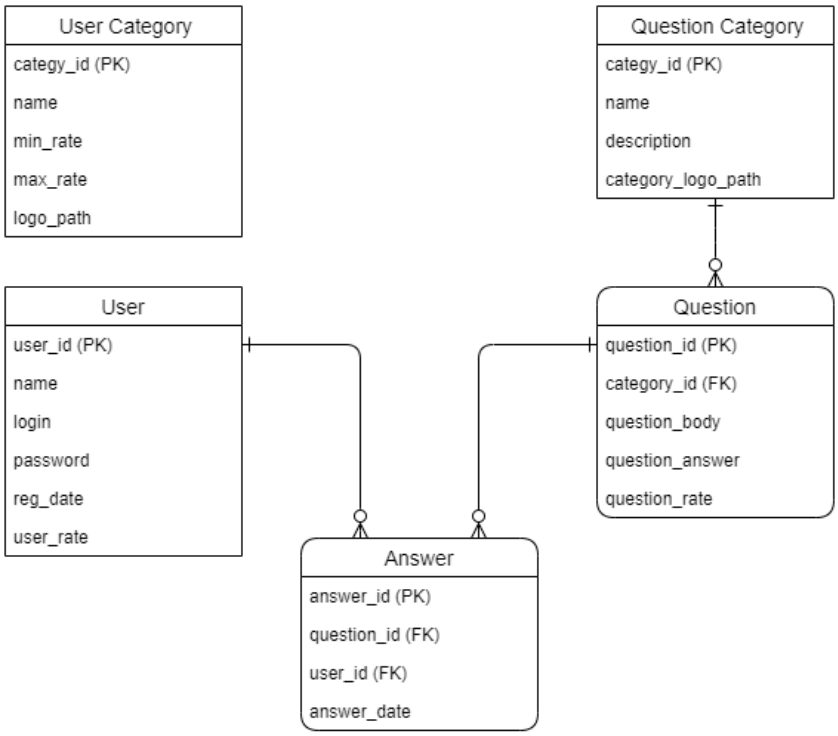

Рисунок 3.1 – ER-діаграма інформаційної системи

3.1.5 Реалізація діаграми варіантів використання

Модель варіантів використання є однією з поведінкових діаграм UML та описує функціональні вимоги до системи у вигляді варіантів використання або прецедентів.

Це гарний інструмент для планування на початкових етапах, бо одразу може дати уявлення про функції системи та зовнішню взаємодію з ними.

Ця діаграма описує які сутності (актори) та яким чином (варіанти використання) взаємодіють із системою.

Дана модель є потужним інструментом та може використовуватися на усіх етапах проектування та розробки системи.

Розглянемо модель використання нашої системи з точки зору користувача готового додатку.

На рис. 3.2 зобразимо усі вимоги до функціоналу, що були описані на етапі постановки завдання (підрозділ 1.7).

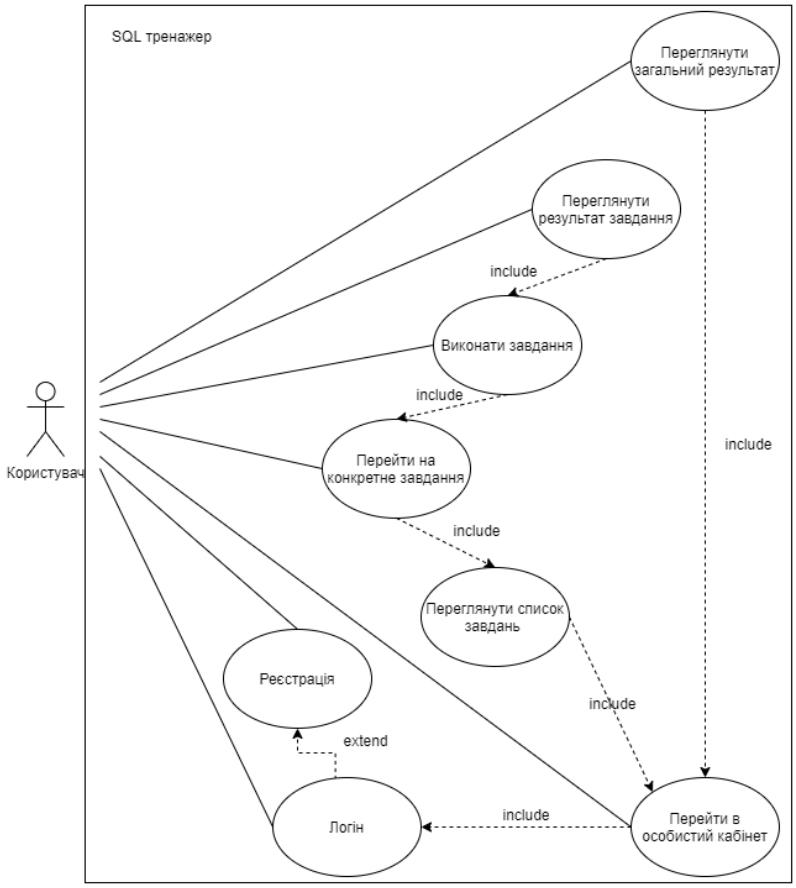

Рисунок 3.2 – Діаграма варіантів використання інформаційної системи

3.1.6 Реалізація моделі IDEF0

Центральним елементом моделі IDEF0 є функція, яка на схемі відображається у вигляді функціонального блоку-прямокутника, всередині якого зазначено дію в формі іменника, який походить від дієслова.

На рис. 3.3 покажемо контекстну діаграму нульового рівня, на якій зобразимо потоки інформації, важливі для нашої системи.

Рисунок 3.3 – IDEF0 нульового рівня

Як бачимо система буде отримувати та обробляти дані користувача та його відповіді на питання (входи системи). Користувач у своїх діях буде обмежений інструкцією користувача та синтаксисом мови SQL (потоки управління). Використовуючи ресурси СКДБ PostgreSQL та логіки, що реалізована як сервер застосунку, (механізми) система надає можливості для перегляду результатів та даних користувача (виходи).

Далі, на рис. 3.4, показана декомпозиція на діаграму першого рівня, що більш детально показує процеси роботи системи. Тут ми можемо побачити яка інформація та як передається під час виконання основних варіантів використання системи, що були описані раніше, у підрозділі 3.1.4.

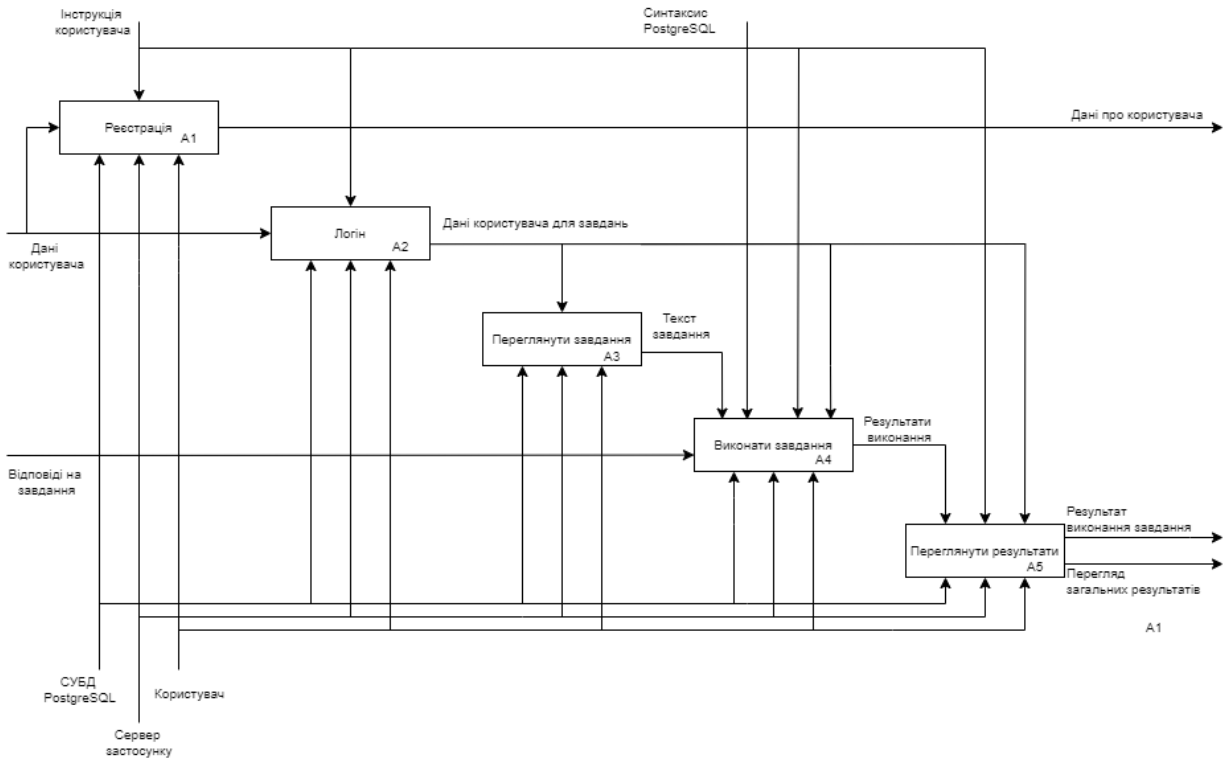

Рисунок 3.4 – IDEF0 першого рівня

3.1.7 Реалізація діаграми класів

Діаграма класів – це один із головних інструментів при проектуванні об’єкно-орієнтованого проекту. Це структурна діаграма UML. Вона показує класи системи, атрибути та методи кожного класу та зв’язки між класами.

Кожен клас на діаграмі позначається у вигляді прямокутника з назвою класу вгорі, полями класу посередині та методами внизу.

Зв’зки між класами позначаються різними типами стрілочок.

Розглянемо основні зв’язки, що використовуються на діаграмі класів.

- асоціація – відповідає наявності деякого зв’язку між класами.
- агрегація – має місце між декількома класами в тому випадку, якщо один з класів є деякою сутність, що включає в себе в якості складових частин інші сутності.
- наслідування – зв’язок, що показує відносини між батьківськими та дочірніми сутностями.

Розпочнемо розробку діаграми класів із визначення того, як система повинна:

- отримувати доступ до даних у сховищі даних;
- збірагати та обробляти дані під час виконання програми;
- обробляти дії користувача;

Обираючи трьохкомпонентну модель додатку [36], ми можемо відділити наступні типи класів:

- класи моделі – класи, що будуть презентувати основні сутності програми (подібно до сутностей в ER моделі). Класи моделі це звичайні класи, що відповідають якійсь сутності програми та містять у собі поля з відповідними атрибутами;
- класи доступу до бази даних – класи, що будуть отримувати необхідні дані зі сховища та перетворювати їх на об’єкти класів моделі;
- класи сервіси – класи, що, по суті, реалізують логіку, правила та вимоги програми.

Використовуючи раніше описані моделі сутностей (рис. 3.1) та варіантів використання (рис. 3.2) можемо зробити висновок, що кожна сутність бази даних отримає свій клас моделі (рис. 3.5) та доступу до бази даних (рис. 3.6), а кожен із варіантів використання буде реалізований за допомогою класів-обробників (рис. 3.7).

Повну діаграму класів разом із позначенням зв’зків наведено на рис. 3.8.

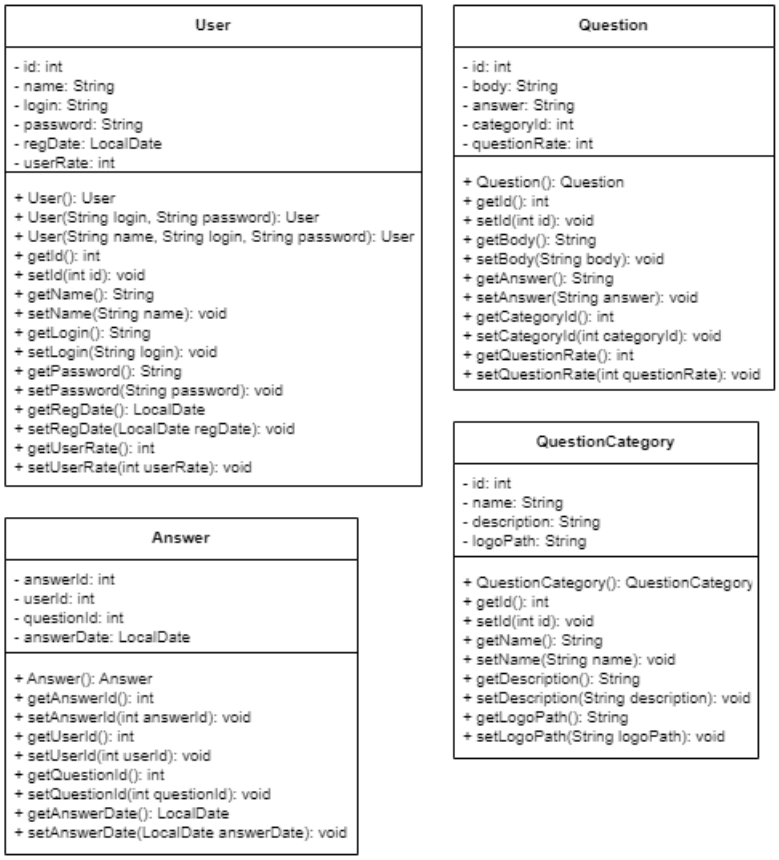

Рисунок 3.5 – Діаграма класів моделі

Рисунок 3.6 – Діаграма класів доступу до бази даних

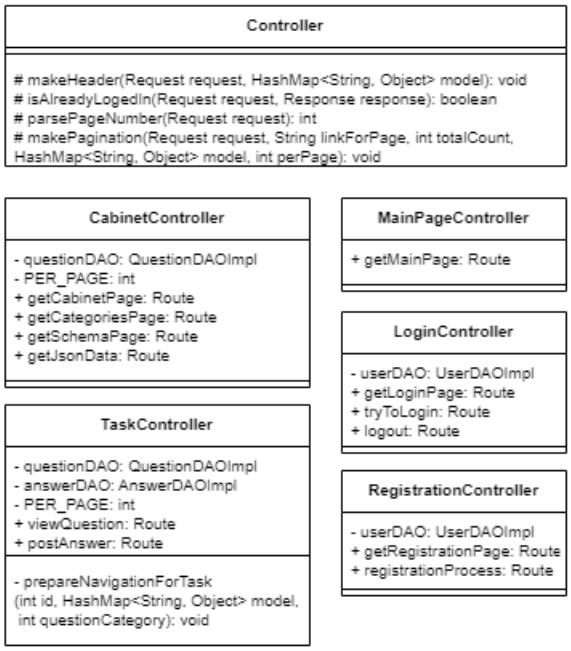

Рисунок 3.7 – Діаграма класів контролерів

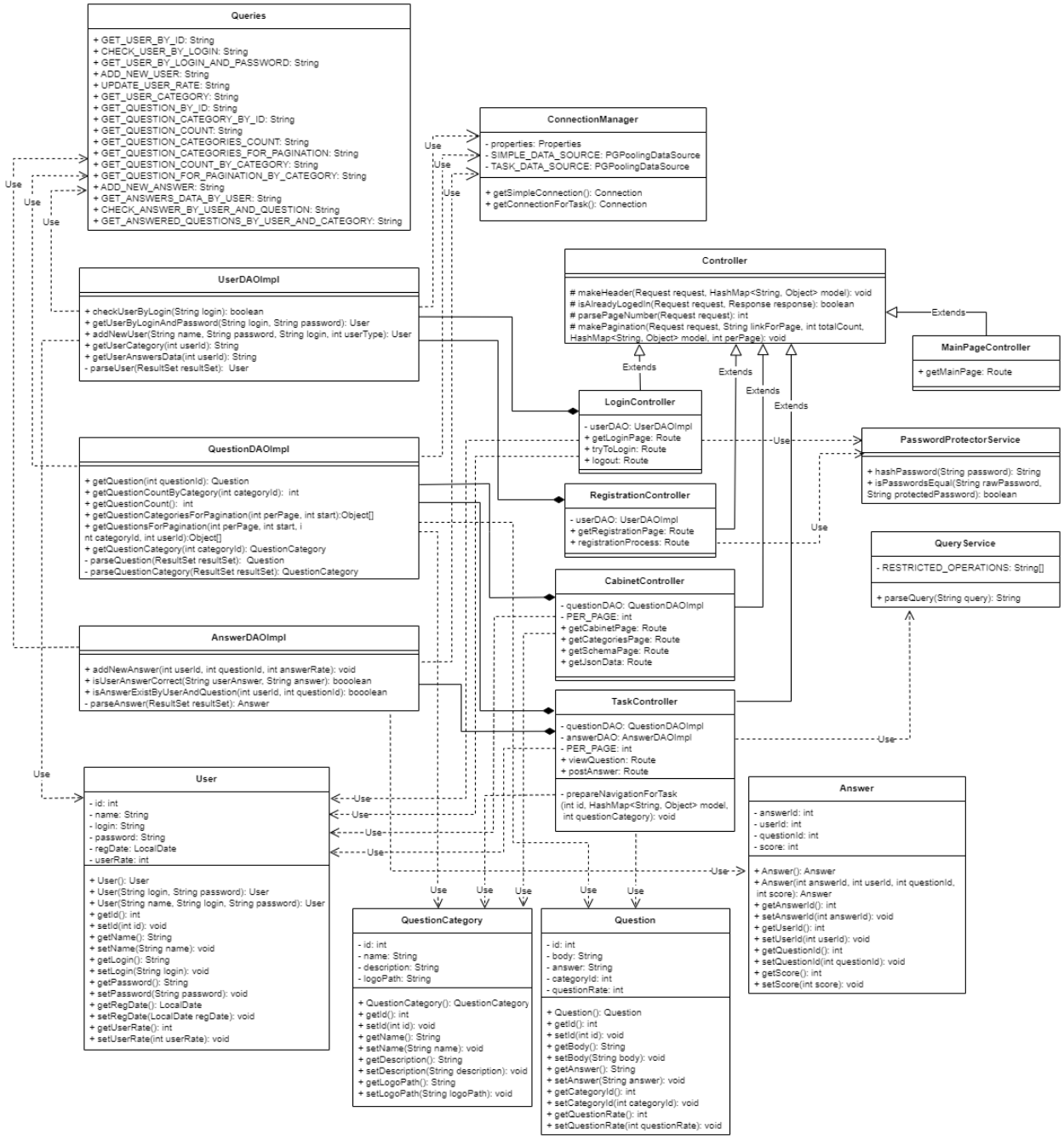

Рисунок 3.8 – Діаграма класів додатку із зв’язками

3.2 Опис та формування вхідних даних

3.2.1 Опис даних тренажеру

У системі, до початку роботи з нею користувача, вже представлена певна кількість інформації. Це дані про категорії користувачів, категорії питань та вже підготовлені питання. Ці дані будуть підготовлені та додані до бази даних перед стартом роботи програми. У якості тренувальної схеми

обрано класичну схему EMP-DEPT [40]. Скрипти ініціалізації та заповнення бази даних наведено у Додатку А.

3.2.2 Опис даних користувача

Також система приймає інформацію від користувача. Це логін, пароль та ім’я користувача, а також відповіді на питання.

Усі ці данні мають наступні обмеження:

- логін – може складатися із цифр та символів, має бути унікальним;
- пароль – може складатися із цифр та символів. Довжина – не менше 8 символів;
- ім’я користувача – може складатися із цифр та символів;
- відповідь користувача – являє собою SQL-запит. Не може містити конструкцій, що можуть завдати шкоди цілосності бази даних (такі як drop, delete, truncate, insert, update). Задля задоволення цієї вимоги відбувається фільтрація тексту запиту за ключовими словами та обмеження прав користувача при виконанні запитів: дозволено виконувати лише операцію вибірки (select) інформації із бази даних.

3.3 Програмна реалізація інформаційної системи

3.3.1 Архітектура додатку та організація коду

Як раніше зазначалося у підрозділі 3.1.7 ми використовуємо трьохкомпонентну модель додатку.

Також ми реалізуємо веб-додаток, тому для ефективної організації коду ми можемо використати шаблон проектування «Модель-Вигляд-Контроллер» (Model-View-Controller, MVC) [37].

Даний шаблон передбачає розділення класів та ресурсів додатку на наступні типи:

- класи-контролери (Controller) – класи, що проводять оброку запитів користувача;

- класи-моделі (Model) – класи, що зберагіють інформацію про сутності програми;
- вигляди (View) – шаблони, що дать змогу користувачеві переглянути результати роботи.

Також щоб не перевантажувати класи контролерів великою кількістю коду, ми можемо використати патерн «Команда» [38] та винести частину коду в класи сервісів.

Загальний варіант архітектури додатку показано на рис. 3.9.

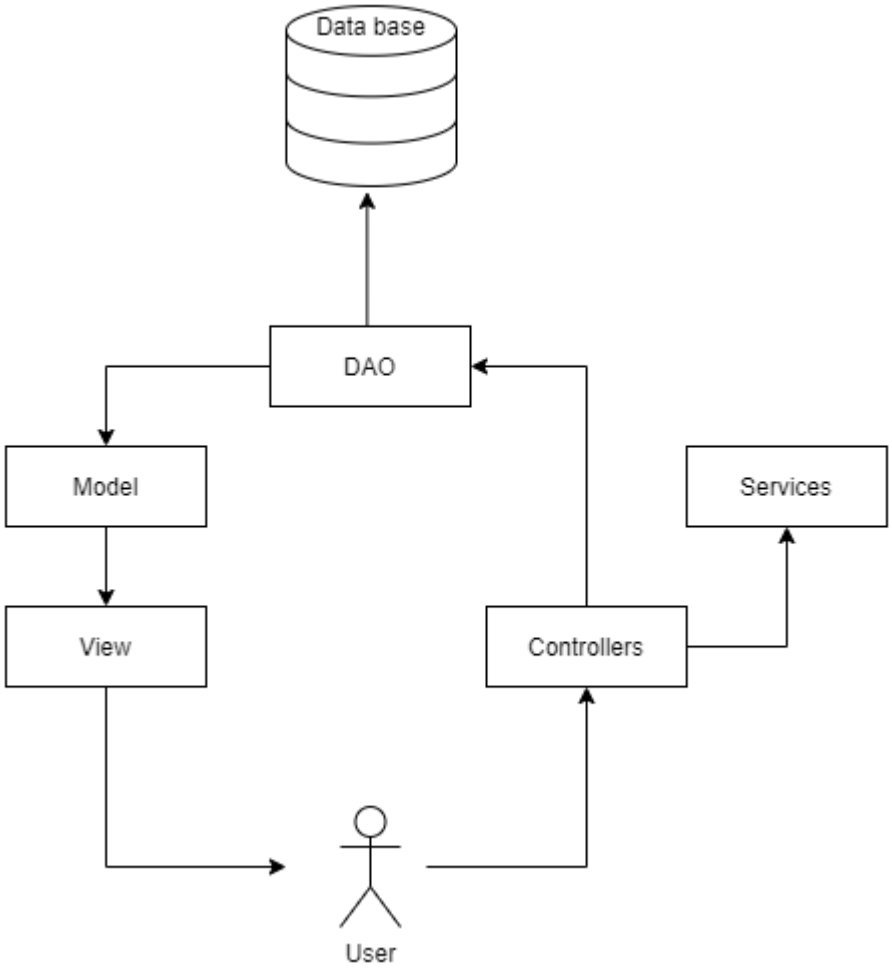

Рисунок 3.9 – Схема архітектури додатку

Використовуючи раніше розроблену діаграму класів, розглянуту вище архітектуру додатку та підхід мови Java до логічної організації класів за допомогою пакетів, розмістимо класи додатку як показано на рис. 3.10.

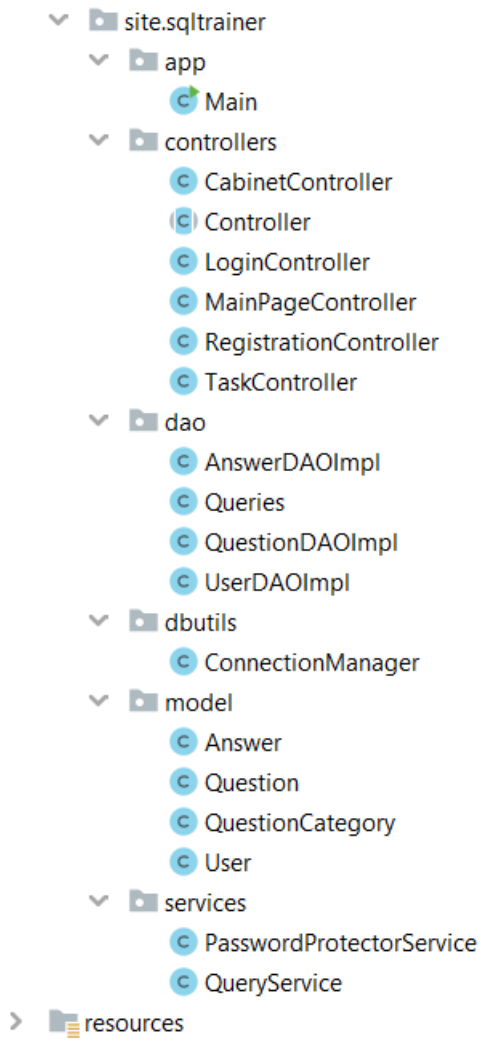

Рисунок 3.10 – Розміщення класів додатку в IDE Intellij IDEA

3.3.2 Короткий опис програмної реалізації

Розглянемо загальний підхід до реалізації функціоналу програми.

Розглянемо приклад реалізації класу User, що представляє собою сутність користувача.

Даний клас містить поля, контруктор без параметрів та методи для встановлення й отримання значень для полів об’єкта класу.

```
public class User {
    private int id;
    private String name;
    private String login;
    private String password;
    private LocalDate regDate;
    private int userRate;
    public User() {
    }

   getters and setters;
}
```

При розробці класів доступу до бази даних ми використаємо стандартні інструменти доступу до дази даних мови Java, а саме JDBC API [39].

По-перше, сформуємо підключення до бази даних у класі ConnectionManager (Додаток Б), де зчитуємо параметри підключення до бази даних та формуємо сам об’єкт підключення.

Для формування запиту до бази даних використаємо підготовлені запити, щоб уникнути SQL-ін’єкції.

Приклад методу, що здійснює операцію додавання нового користувача в базу даних наведено нижче.

```
public User addNewUser(String name, String password, String login) {
    User user = null;
    try (Connection connection = ConnectionManager.getSimpleConnection();
         PreparedStatement insertStatement =
connection.prepareStatement(Queries.ADD_NEW_USER);
         PreparedStatement selectStatement =
connection.prepareStatement(Queries.GET_USER_BY_LOGIN_AND_PASSWORD);) {
        insertStatement.setString(1, name);
        insertStatement.setString(2, login);
        insertStatement.setString(3, password);
        insertStatement.setDate(4, Date.valueOf(LocalDate.now()));
        insertStatement.execute();
        selectStatement.setString(1, login);
        ResultSet resultSet = selectStatement.executeQuery();
        resultSet.next();
        user = parseUser(resultSet);
    } catch (SQLException e) {
        LOGGER.error("Error while adding new user ", e);
    }
    return user;
}
```

У класах контролерах ми вже використаємо можливості фрейморку Spark та лямба-виразів.

Тут ми формуємо обробники конкретних запитів користувача, що надходять із веб-браузера на сервер додатку. Інформацію про те, які дані були надіслані до сервера користувачем ми отримуємо завдяки об’єкту класу Request.

Далі ми формуємо власне тимчасове сховище даних за допомогою хеш мапи з даними, що необхідно буде віддати користувачеві.

У більшості випадків ми також перевіряємо чи авторизований користувач для виконання тієї чи іншої операції.

Для виконання деяких операцій ми можемо викликати методи класів-сервісів.

Після цього ми формуємо вхідні дані для запиту до бази даних та викликаємо методи класів доступу до бази даних.

Далі ми формуємо відповідь для користувача, отримавши та опрацювавши необхідні дані з бази, та кладемо її до нашого тимчасового сховища, що буде використано при створенні сторінки відповіді для користувача.

Нижче наведено приклад методу обробки запиту користувача на проходження завдання у TaskController.

```
    public final Route postAnswer = ((request, response) -> {
        HashMap<String, Object> model = new HashMap<>();
        User user = request.session().attribute("user");
        if (user == null) {
            model.put("message", "Ви не авторизовані");
            return new ThymeleafTemplateEngine().render(new
ModelAndView(model, "index"));
        }
        String query = request.queryParams("query_body");
        Question question = request.session().attribute("question");
        String parsedQuery = QueryService.parseQuery(query);
        prepareNavigationForTask(question.getId(), model,
question.getCategoryId());
        model.put("user_answer", query);
        if (parsedQuery == null) {
            model.put("message", "Виконувати дозволено лише Select'и");
        } else {
            if (answerDAO.isAnswerExistByUserAndQuestion(user.getId(),
question.getId())) {
                model.put("message", "Ви вже виконали це завдання, спробуйте
наступне.");
            } else {
                boolean isAnswerCorrect =
answerDAO.isUserAnswerCorrect(parsedQuery, question.getAnswer());
                if (isAnswerCorrect) {
                    model.put("message", "Правильно, перейдіть до іншого
питання.");
                    answerDAO.addNewAnswer(user.getId(), question.getId(),
question.getQuestionRate());
                } else {
                    model.put("message", "Будь ласка, спробуйте ще раз.");
                }
            }
        }
        model.put("user", user);
        model.put("question", question);
        return new ThymeleafTemplateEngine().render(new ModelAndView(model,
"task"));
    });
```

Для формування динамічного вмісту сторінки використовуємо Thymeleaf. Цей двигун шаблонів підключається на HTML-сторінку за допомогою посилання xmlns:th="http://www.thymeleaf.org", а потім ми можемо використовувати його конструкції в стандартних тегах для обробки даних, що були передані з класу-контрелеру.

Також тут ми використовуємо класи Bootstrap для формування адаптивного дизайну.

Приклад для виведення тексту питання наведено у наступному коді:

```
<div th:object="${question}" class="row">
    <div class="col-md-9 mb-20 mt-5">
        <div class="single-recipe hover-effect-one fix">
            <div class="recipe-image box-hover recipe-shadow">
                <div class="row">
                    <div class="col-md-12 text-left">
                        <h5><u>Завдання:</u></h5>
                        <br>
                        <h6 th:text="${question.body}"></h6>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

Опишемо загальний підхід до реалізації автоматичного тестування додатку.

Перш за все ми зробимо імітацію серверу веб-додатку за допомогою засобів Spark та HttpClients. Даний сервер буде викликати при обробці запитів ті самі класи, що й реальний додаток.

```
public static class SparkServer {
    private static final LoginController LOGIN_CONTROLLER = new
LoginController();
    public static void main(String[] args) {
        Spark.port(8888);
        Spark.threadPool(1000, 1000, 60000);
    }

    //login
    public static void getLoginPage() {
        Spark.get("/login", LOGIN_CONTROLLER.getLoginPage);
    }
}
```

Далі створимо методи тестів, використовуючи анотацію @Test. У тілі цих методів ми будемо формувати HTTP-запити, виконувати їх на нашому

раніше підготовленому сервері та порівнювати результати виконання із підготовленими даними.

Інструменти HttpClient дозволяють отримати повну інформацію про виконання запиту: заголовки запиту, статус-коди відповідей, тіло та заголовки відповіді тощо.

Але поки що для виконання тестів нам буде достатньо порівняти коди відподей запитів: якщо запит виконано успішно, то код відповіді буде дорівнювати 200 і тест буде пройдено. Приклад реалізації методу перевірки отримання сторінки логіну наведено нижче, а повний код виконання тестування для основних варіантів використання з позитивним для користувача результатом наведено у класі TestRun (Додаток Б).

```
@Test
public void testGetLoginPage() throws IOException {
    SparkServer.getLoginPage();
    CloseableHttpClient httpClient = HttpClients.custom().build();
    HttpGet httpGet = new HttpGet("http://localhost:8888/login");
    CloseableHttpResponse response = httpClient.execute(httpGet);
    int statusCode = response.getStatusLine().getStatusCode();
    assertEquals(200, statusCode);
}
```

У результаті написання тестів, було покрито приблизно 70 відсотків коду (рис. 3.11 та 3.12), що є досить високим показником. Додаткового покриття потребує код, що виконує логіку з негативним для користувача результатом (невдала спроба виконання завдання, логіну тощо).

GraduateWork

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
site.sqltrainer.controllers		70%		46%	33	62	70	253	2	29	0	6
site.sqltrainer.dao		82%		71%	7	32	47	241	3	25	1	4
site.sqltrainer.app		0%		0%	9	9	37	37	8	8	2	2
site.sqltrainer.model		68%		n/a	11	44	23	74	11	44	1	4
site.sqltrainer.dbutils		70%		n/a	1	4	7	37	1	4	0	1
site.sqltrainer.services		89%		66%	4	9	4	15	2	6	0	2
Total	754 of 2,645	71%	43 of 88	51%	65	160	188	657	27	116	4	19

Рисунок 3.11 – Результати покриття коду тестами в Jacoco

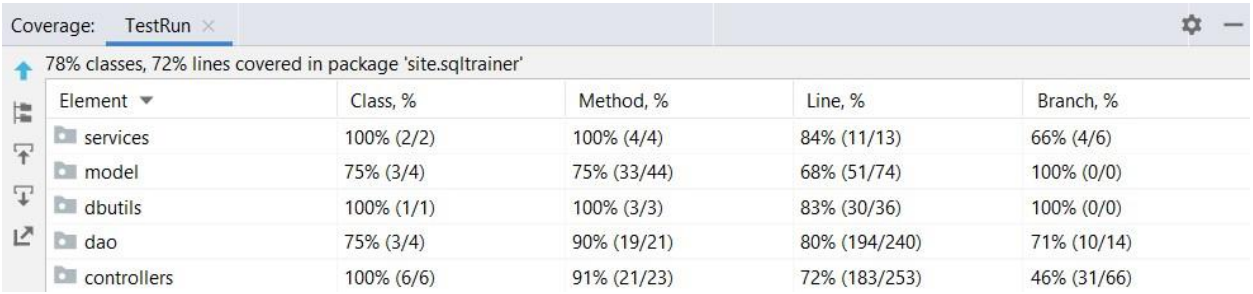

Рисунок 3.12 – Результати покриття коду тестами в Intellij IDEA

3.4 Аналіз роботи програми

Розглянемо приклади виконаної роботи. Проглянемо реалізацію кожного з варіантів використання:

- Перегляд головної сторінки (рис 3.13);

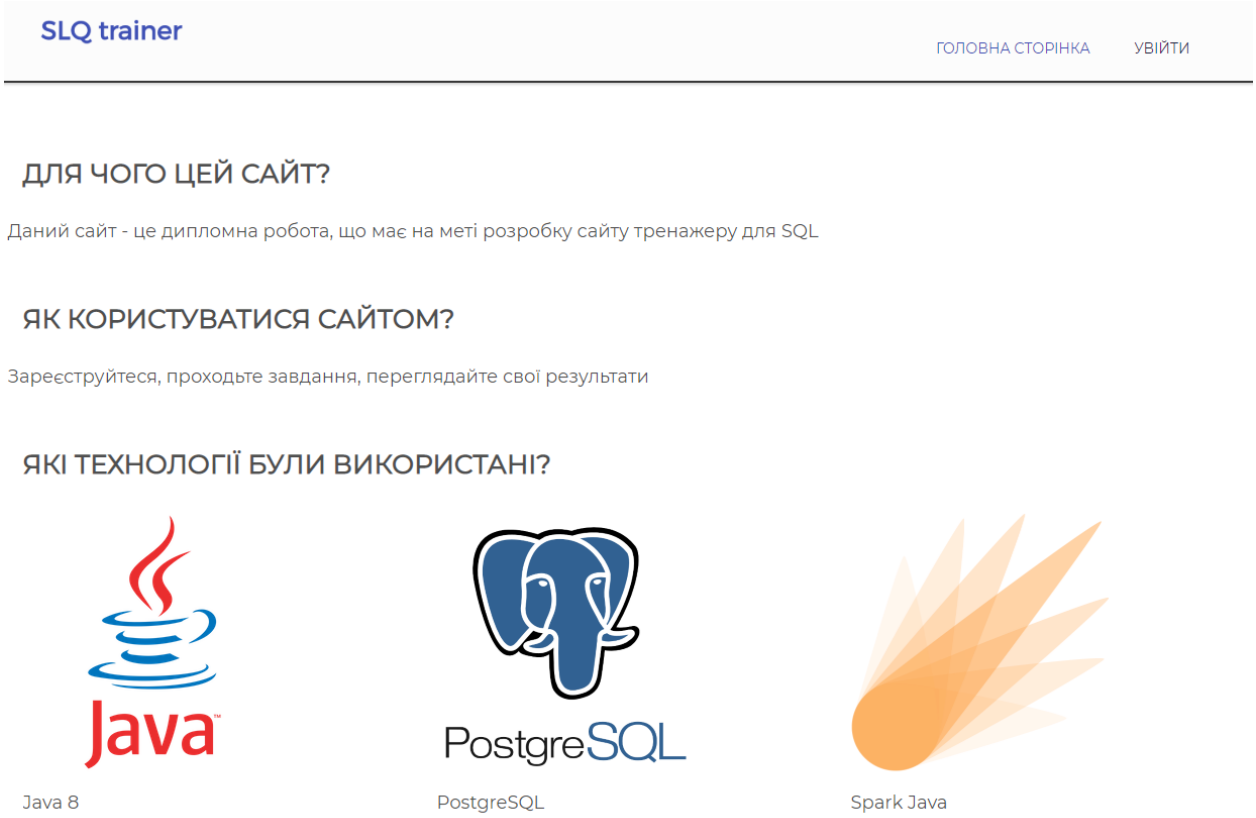

Рисунок 3.13 – Головна сторінку додатку

- Реєстрація (рис. 3.14);

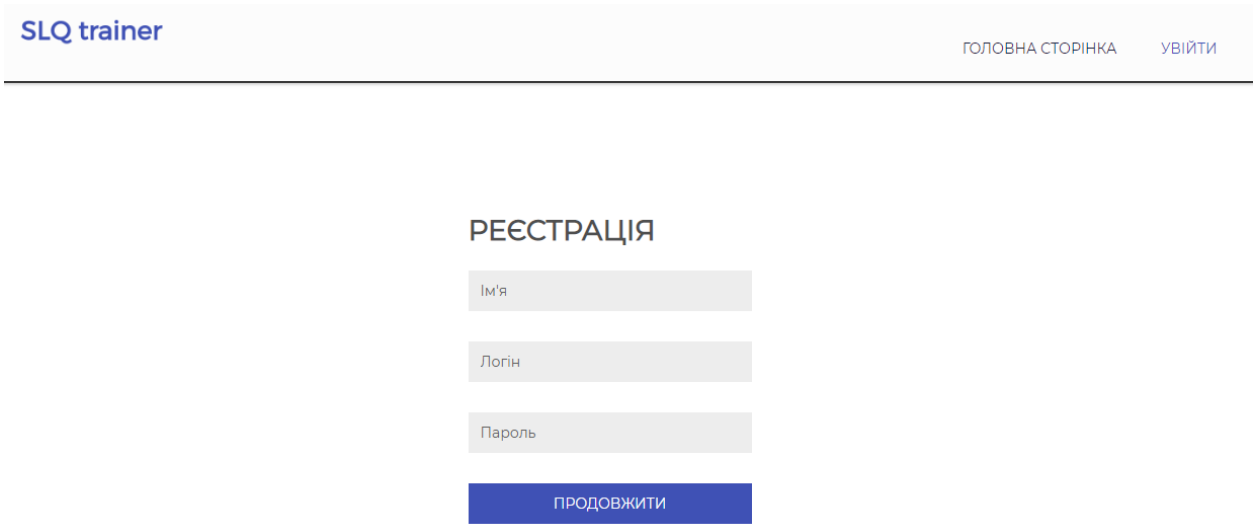

Рисунок 3.14 – Сторінка реєстрації

- Авторизація (рис 3.15);

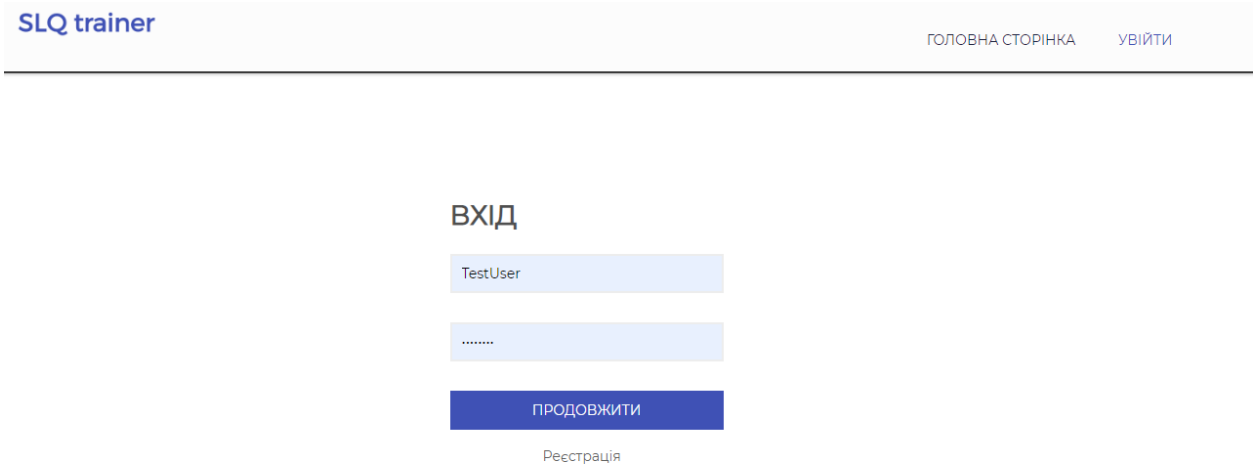

Рисунок 3.15 – Сторінка авторизації

- Перехід до особистого кабінету та перегляд загальних результатів (рис. 3.16);

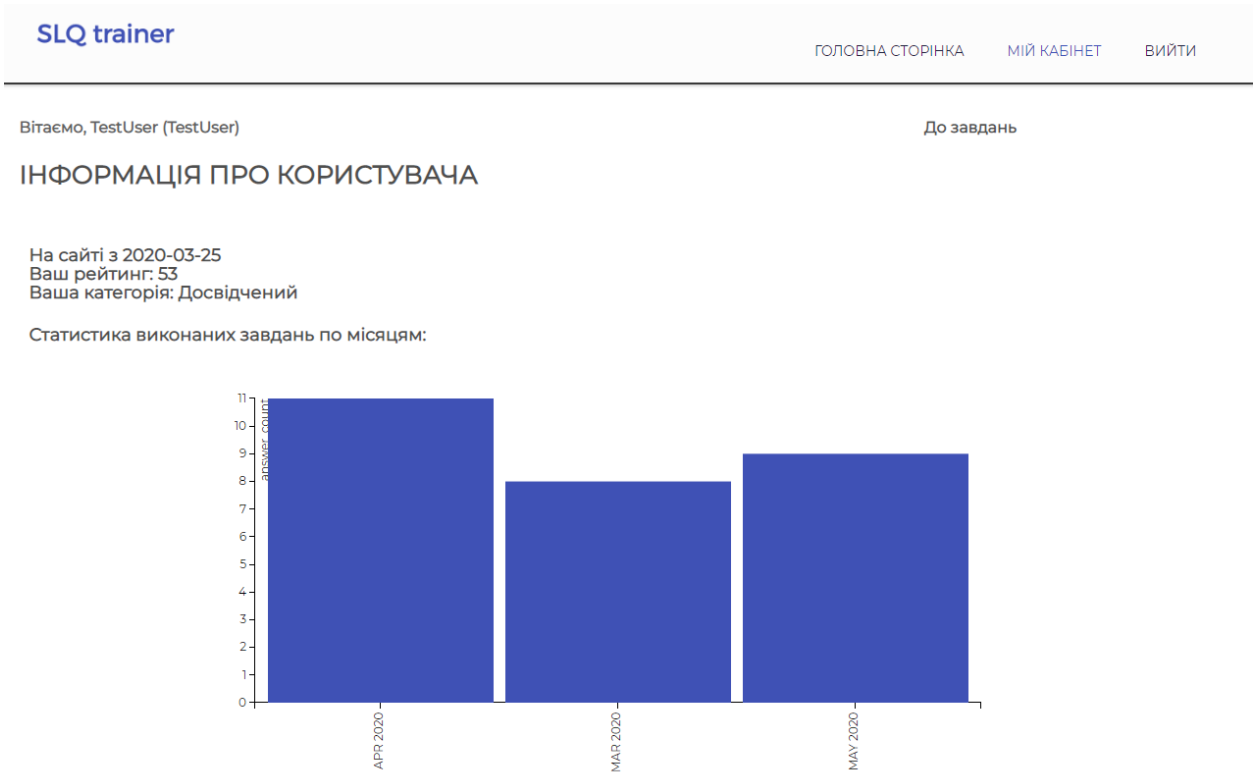

Рисунок 3.16 – Сторінка особистого кабінету

- Виконання завдання (рис. 3.17);

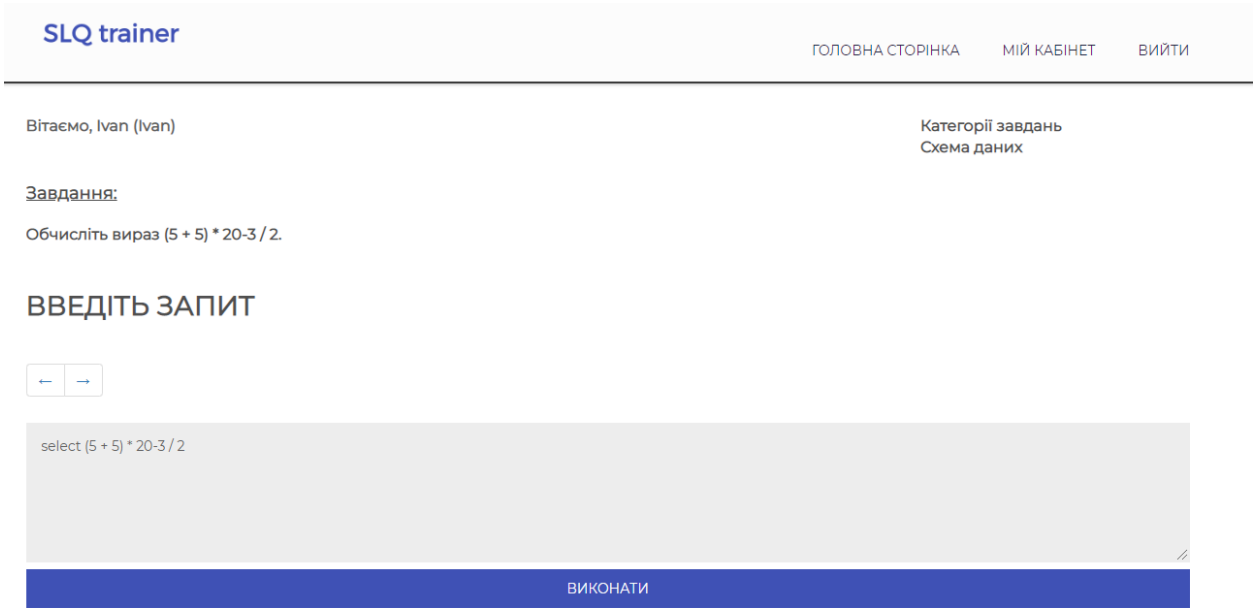

Рисунок 3.17 – Сторінка виконання завдання

- Перегляд результатів виконання завдання (рис. 3.18 та 3.19);

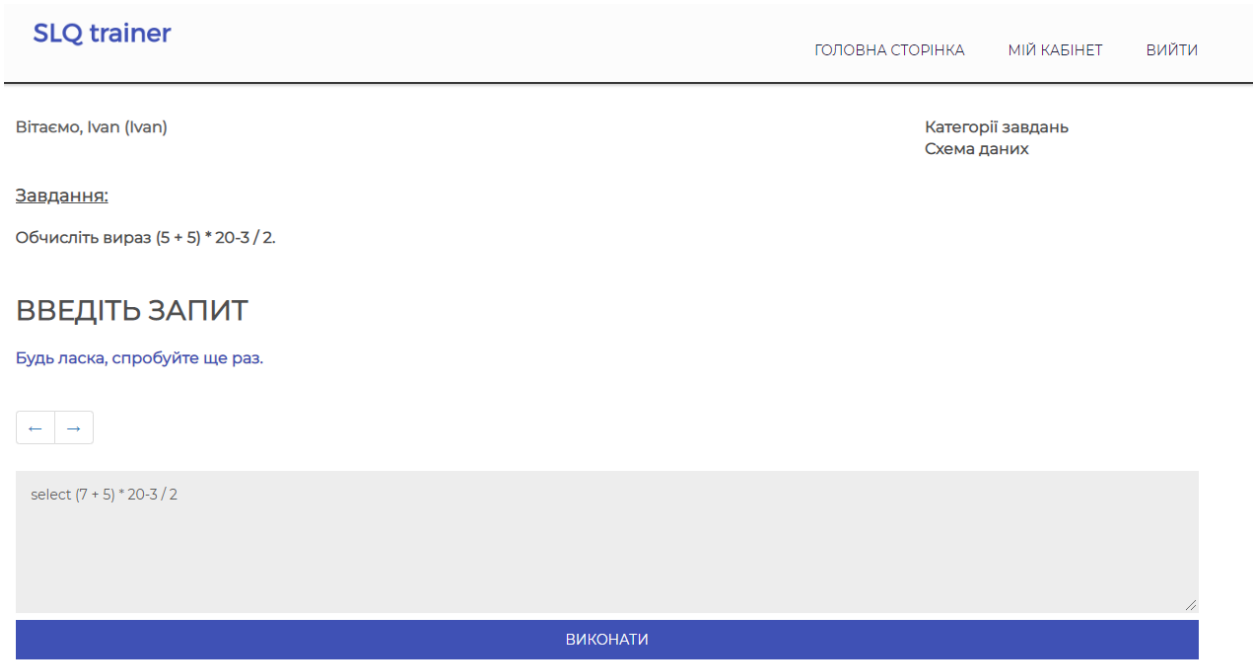

Рисунок 3.18 – Приклад негативного результату виконання завдання

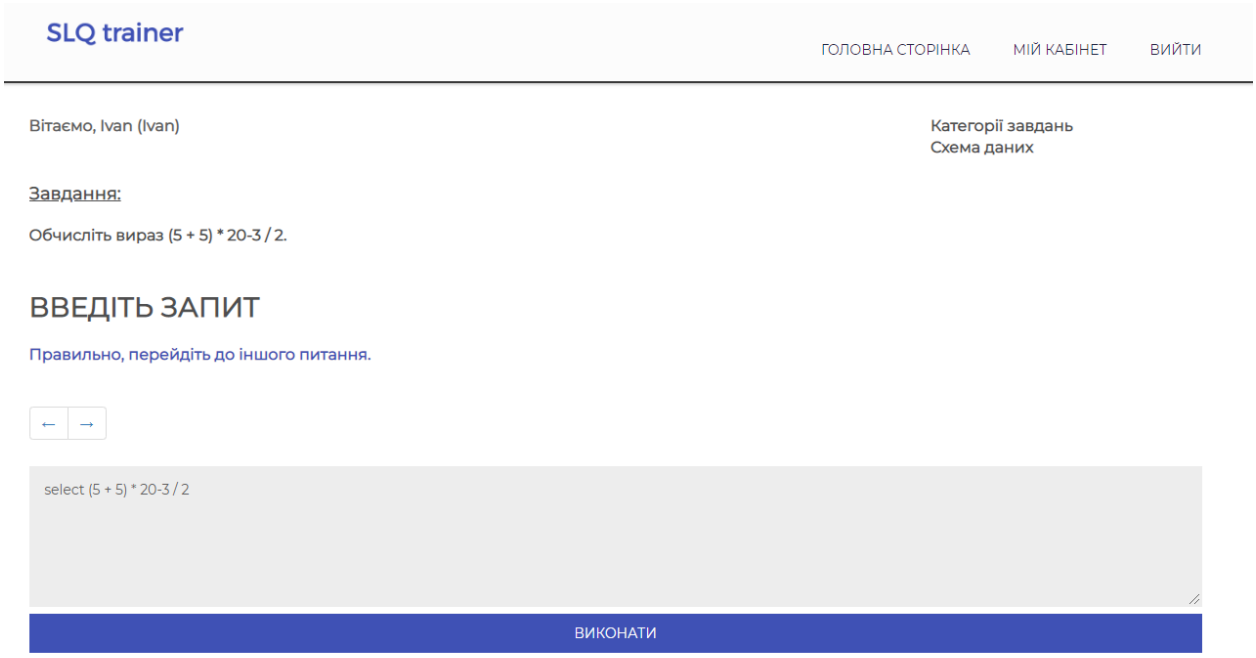

Рисунок 3.19 – Приклад позитивного результату виконання завдання

- Мобільна версія (рис. 3.20 та 3.21);

Рисунок 3.20 – Мобільна версія сторінки виконання завдання

Рисунок 3.21 – Мобільна версія сторінки особистого кабінету

ВИСНОВКИ

Під час роботи було підтверджено те, що знання та використання мови SQL є затребуваними на сучасному ринку IT-вакансій, особливо у сфері веб-розробки, та що широке розповсюдження засобів навчання SQL є актуальним.

Було розглянуто способи масового навчання та проаналізовано один із найпопулярніших та найдієвіших – онлайн-платформу. У ході інформаційного огляду було розглянуто найпопулярніші в українському медіа-просторі навчальні платформи «Stepik» та «Prometheus» та SQL-тренажери W3School та SoloLearn.

Також було вирішено розробляти платформу у вигляді веб-додатку. Для цього було розглянуто сучасні підходи до побудови та організації веб-додатків та поточні тенденції у сфері веб-розробки, а саме:

- використання клієнтської та серверної частини веб-додатку;
- надання сайту як у десктопній версії, так і в мобільній;
- написання автоматичних тестів для підвищення якості додатку.

Під час реалізації було спочатку проведено проектування інформаційної системи, у яході якого було розроблено:

- ER-діаграму;
- діаграму варіантів використання;
- модель IEDF0;
- діаграму класів.

На основі спроектованих діаграм було розроблено сайт із використанням мови Java, веб-фреймворку Spark, СКДБ PostgreSQL та інших технологій.

Також було проведено тестування додатку за допомогою автоматичного тестування з використанням інструментів JUnit 5, JaCoCo та HttpClient.

Під час роботи було поглиблено знання мови Java, мови SQL діалекту PostgreSQL. Вивчено нові методи роботи з форматом JSON у СКДБ PostgreSQL та роботу з візуалізацією даних за допомогою бібліотеки D3.js.

Розроблений додаток також має перспективи для подальшого розвитку та втілення нових особливостей:

- Доповнення елементів інтерактивності:
  - сповіщення;
  - система досягнень;
  - глобальний рейтинг користувачів.
- Розробка онлайн чат-ботів для підтримки користувачів;
- Розробка системи комунікаццїї для користувачів:
  - повідомлення;
  - оголошення;
  - обговорення;
  - коментарі.
- Збільшення відсотку покриття коду тестами та покращення самих методів тестування.

У результаті роботи було отримано SQL-тренажер з можливостями реєстрації, авторизації, перегляду та виконання завдань, що працює у всіх сучасних браузерах, має адаптивний дизайн та мобільну версію та є зручним і зрозумілим у використанні.

СПИСОК ЛІТЕРАТУРИ

1. DB-Engines Ranking [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://db-engines.com/en/ranking (дата звернення 01.05.2020) – Назва з екрана.
2. Why is SQL the most important skill to learn? [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://tableplus.com/blog/2018/08/why-sql-is-the-most-important-skill-to-learn.html (дата звернення 01.05.2020) – Назва з екрана.
3. Вакансии | DOU [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://jobs.dou.ua/vacancies/ (дата звернення 03.05.2020) – Назва з екрана.
4. Java. Базовый курс [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://stepik.org/course/187/syllabus (дата звернення 03.05.2020) – Назва з екрана.
5. Основи програмування на Java [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://courses.prometheus.org.ua/courses/EPAM/JAVA101/2016_T2/about (дата звернення 04.05.2020) – Назва з екрана.
6. SQL Tutorial [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.w3schools.com/sql/ (дата звернення 05.05.2020) – Назва з екрана.
7. Learn SQL [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://play.google.com/store/apps/details?id=com.sololearn.sql&hl=en (дата звернення 05.05.2020) – Назва з екрана.
8. Difference between Website and Web Application [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.guru99.com/difference-web-application-website.html (дата звернення 05.05.2020) – Назва з екрана.

9. What is a Web Application? [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://blog.stackpath.com/web-application/ (дата звернення 05.05.2020) – Назва з екрана.
10. Technology Stack for Web Application Development [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://rubygarage.org/blog/technology-stack-for-web-development (дата звернення 07.05.2020) – Назва з екрана.
11. Choosing a technology stack for web application development [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://light-it.net/blog/choosing-a-technology-stack-for-web-application-development/ (дата звернення 07.05.2020) – Назва з екрана.
12. Web Applications Development Trends 2020 – The Top Ten [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://eleks.com/blog/web-applications-development-trends-2020/ (дата звернення 08.05.2020) – Назва з екрана.
13. 8 Web Development Trends Every CTO Should Be Ready for in 2020 [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.cleveroad.com/blog/web-development-trends (дата звернення 10.05.2020) – Назва з екрана.
14. 15 Top web development trends in 2020 [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://lanars.com/blog/top-web-development-trends (дата звернення 10.05.2020) – Назва з екрана.
15. Java Platform Standard Edition 8 Documentation [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://docs.oracle.com/javase/8/docs/ (дата звернення 11.05.2020) – Назва з екрана.
16. 15+ Top Reasons to Choose Java for Backend Development [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.yourteaminindia.com/blog/java-backend-development/ (дата звернення 11.05.2020) – Назва з екрана.

17. TIOBE Index for May 2020 [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.tiobe.com/tiobe-index/ (дата звернення 11.05.2020) – Назва з екрана.
18. Spark Framework: An expressive web framework for Kotlin and Java [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: http://sparkjava.com/ (дата звернення 12.05.2020) – Назва з екрана.
19. Jetty - Servlet Engine and Http Server [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.eclipse.org/jetty/ (дата звернення 12.05.2020) – Назва з екрана.
20. Thymeleaf [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.thymeleaf.org/ (дата звернення 12.05.2020) – Назва з екрана.
21. PostgreSQL: The World's Most Advanced Open Source Relational Database [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.postgresql.org/ (дата звернення 12.05.2020) – Назва з екрана.
22. Advantages of PostgreSQL [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.cybertec-postgresql.com/en/postgresql-overview/advantages-of-postgresql/ (дата звернення 12.05.2020) – Назва з екрана.
23. PostgreSQL vs. MySQL: A 360-degree Comparison [Syntax, Performance, Scalability and Features] [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.enterprisedb.com/blog/postgresql-vs-mysql-360-degree-comparison-syntax-performance-scalability-and-features (дата звернення 12.05.2020) – Назва з екрана.
24. D3.js - Data-Driven Documents [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://d3js.org/ (дата звернення 12.05.2020) – Назва з екрана.
25. Bootstrap The most popular HTML, CSS, and JS library in the world. [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу:

https://getbootstrap.com/docs/4.5/getting-started/introduction/ (дата звернення 12.05.2020) – Назва з екрана.

26. Junit 5 [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://junit.org/junit5/ (дата звернення 12.05.2020) – Назва з екрана.
27. EclEmma - JaCoCo Java Code Coverage Library [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.eclemma.org/jacoco/ (дата звернення 12.05.2020) – Назва з екрана.
28. HttpClient Tutorial [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://hc.apache.org/httpcomponents-client-4.3.x/tutorial/html/index.html (дата звернення 12.05.2020) – Назва з екрана.
29. Top 10+ Best Java IDEs & Online Java Compilers [2020 Rankings] Bootstrap · The most popular HTML, CSS, and JS library in the world. [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.softwaretestinghelp.com/best-java-ide-and-online-compilers/ (дата звернення 12.05.2020) – Назва з екрана.
30. Why software design is important? [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://hackernoon.com/why-software-design-is-important-9ecbea883bbb (дата звернення 15.05.2020) – Назва з екрана.
31. Advantages And Disadvantages Of ER Model In DBMS [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://pctechnicalpro.blogspot.com/2017/04/advantages-disadvantages-er-model-dbms.html (дата звернення 15.05.2020) – Назва з екрана.
32. Key benefits of using entity relationship diagrams [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://pcdreams.com.sg/key-benefits-of-using-entity-relationship-diagrams/(дата звернення 15.05.2020) – Назва з екрана.

33. IDEF0 - Part 1 (understanding it) [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: http://www.syque.com/quality_tools/tools/Tools19.htm (дата звернення 15.05.2020) – Назва з екрана.
34. Introduction to unified modeling language [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.uml.org/what-is-uml.htm (дата звернення 15.05.2020) – Назва з екрана.
35. Why the Software Industry Has a Love-Hate Relationship with UML Diagrams [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://creately.com/blog/diagrams/advantages-and-disadvantages-of-uml/ (дата звернення 15.05.2020) – Назва з екрана.
36. 3-Tier Architecture: A Complete Overview [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.jinfonet.com/resources/bi-defined/3-tier-architecture-complete-overview/ (дата звернення 15.05.2020) – Назва з екрана.
37. What is MVC, and how is it like a sandwich shop? [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.freecodecamp.org/news/simplified-explanation-to-mvc-5d307796df30/ (дата звернення 21.05.2020) – Назва з екрана.
38. Команда [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://refactoring.guru/uk/design-patterns/command (дата звернення 21.05.2020) – Назва з екрана.
39. Lesson: JDBC Basics [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html (дата звернення 21.05.2020) – Назва з екрана.
40. Example EMP and DEPT tables. Classic Oracle tables. [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://livesql.oracle.com/apex/livesql/file/content_O5AEB2HE08PYEPTGCFLZU9YCV.html (дата звернення 21.05.2020) – Назва з екрана.

ДОДАТОК А. СКРИПТИ ЗАПОВНЕННЯ БАЗИ ДАНИХ

У даному додатку наведено скрипти створення та заповнення бази даних.

Заповнення бази даних додатку.

```
drop table answers;
drop table questions;
drop table users;
drop table user_categories;
drop table question_categories;

create table user_categories (
      category_id INTEGER NOT NULL UNIQUE,
      name VARCHAR(300) NOT NULL,
      min_rate INTEGER NOT NULL,
      max_rate INTEGER NOT NULL,
      logo_path VARCHAR(300) NOT NULL,

      primary key (category_id)
);

create table users (
      user_id SERIAL NOT NULL UNIQUE,
      name VARCHAR(300) NOT NULL,
      login VARCHAR(300) NOT NULL UNIQUE,
      password VARCHAR(300) NOT NULL,
      reg_date date NOT NULL,
      user_rate INTEGER NOT NULL DEFAULT 0,

      primary key (user_id, user_category)
);

create table question_categories (
      category_id INTEGER NOT NULL UNIQUE,
      name VARCHAR(300) NOT NULL,
      description VARCHAR(1000) NOT NULL,
      category_logo_path VARCHAR(300) NOT NULL,
      primary key (category_id)
);

create table questions (
      question_id SERIAL NOT NULL UNIQUE,
      category_id INTEGER NOT NULL,
      question_body VARCHAR(300) NOT NULL,
      question_answer VARCHAR(300) NOT NULL,
      question_rate INTEGER,

      primary key (question_id, category_id)
);

create table answers (
      answer_id SERIAL NOT NULL UNIQUE,
      user_id INTEGER NOT NULL,
      question_id INTEGER NOT NULL,
      answer_date date NOT NULL,
      primary key (answer_id, user_id, question_id)
);


alter table questions add foreign key (category_id) references
question_categories (category_id) on update restrict on delete restrict;
alter table answers add foreign key (user_id) references users (user_id) on
update restrict on delete restrict;
```

```
alter table answers add foreign key (question_id) references questions
(question_id) on update restrict on delete restrict;

insert into user_categories (category_id, name, min_rate, max_rate,
logo_path) values (1, 'Початківець', 0, 50, 'img/');
insert into user_categories (category_id, name, min_rate, max_rate,
logo_path) values (2, 'Досвідчений', 51, 100, 'img/');

insert into question_categories (category_id, name, description,
category_logo_path) values (1, 'Проста вибірка', 'Одне з найпоширеніших
завдань під час роботи з базою даних - це вибирати дані з таблиць за
допомогою оператора SELECT. Оператор SELECT є одним із найскладніших
операторів у PostgreSQL. У ньому є багато клауз, які можна використовувати
для формування гнучких запитів.', 'img/category/category_1.png');
insert into question_categories (category_id, name, description,
category_logo_path) values (2, 'Обмеження вибірки', 'У клаузі WHERE
використовується умова для фільтрації рядків, повернених із оператора SELECT.
Умова повинна оцінюватися як істинна, хибна чи невідома. Запит повертає
рядки, які задовольняють умові в пункті WHERE.',
'img/category/category_2.png');
insert into question_categories (category_id, name, description,
category_logo_path) values (3, 'Підзапити', 'Підзапит - це запит, вкладений у
інший запит, такий як SELECT, INSERT, DELETE та UPDATE. У цьому тренінгу ми
зосереджуємось лише на операторі SELECT.', 'img/category/category_3.png');
insert into question_categories (category_id, name, description,
category_logo_path) values (4, 'SET операції', 'Дана категорія знаходиться в
розробці, завдання з''являться пізніше.', 'img/category/category_4.png');
insert into question_categories (category_id, name, description,
category_logo_path) values (5, 'Вибірки з декількох таблиць', 'Дана категорія
знаходиться в розробці, завдання з''являться пізніше.',
'img/category/category_5.png');
insert into question_categories (category_id, name, description,
category_logo_path) values (6, 'Агрегатні функції', 'Дана категорія
знаходиться в розробці, завдання з''являться пізніше.',
'img/category/category_6.png');

insert into questions (question_body, category_id, question_answer,
question_rate) values ('Виведіть усі записи з таблиці emp', 1, 'select * from
emp', 1);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Виведіть усі записи з таблиці dept', 1, 'select *
from dept', 1);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Виведіть усі назви відділів із таблиці dept', 1,
'select dname from dept', 1);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Виведіть усі посади працівників без повторень', 1,
'select distinct job from emp', 1);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Виведіть усі назви відділів та їх локації з таблиці
dept', 1, 'select dname, loc from dept', 1);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Виведіть усі міста, де розташовані відділи, без
повторень', 1, 'select distinct loc from dept', 1);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Обчисліть вираз (5 + 5) * 20-3 / 2.', 1, 'select (5 +
5) * 20-3 / 2', 1);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Складіть запит, який поверне для всіх співробітників
ім''я співробітника і розмір його відрахувань в пенсійний фонд. Розмір
відрахувань дорівнює 6% від зарплати.', 1, 'select ename, sal*0.06 from emp',
1);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Складіть запит, який виведе інформацію про всіх
```

```
співробітників в форматі Ім''я-Посада (ім''я дефіс посада)', 1, 'select ename
|| '' - '' || job from emp', 1);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Складіть запит, який виведе інформацію про всіх
відділах в форматі Розташування->Назва', 1, 'select loc || ''->'' || dname
from dept', 2);

insert into questions (question_body, category_id, question_answer,
question_rate) values ('Виведіть імена співробітників на посаді клерк', 2,
'select ename from emp where job = ''CLERK''', 1);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Виведіть ім''я та посаду співробітників, зарплата
яких менше ніж 1500', 2, 'select ename, job from emp where sal < 1500', 1);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Виведіть номер та назву відділів, що розташовані в
Нью-Йорку', 2, 'select deptno, dname from dept where loc = ''NEW_YORK''', 1);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Виведіть імена співробітників у яких є премія', 2,
'select ename from emp where comm is not null', 2);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Виведіть посаду співробітників KING та FORD', 2,
'select job from emp where ename in(''KING'', ''FORD'')', 2);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Виведіть ім''я та посаду співробітників, у яких немає
начальника', 2, 'select ename, job from emp where mgr is null', 2);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Виведете імена і з/п співробітників для яких з/п в
межах $ 1500 і $ 2000.', 2, 'select ename, sal from emp where sal between
1500 and 2000', 2);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Виведіть імена, начальників та премію співробітників,
у яких є начальник, але немає премії', 2, 'select ename, mgr, comm from emp
where mgr is not null and comm is null', 2);

insert into questions (question_body, category_id, question_answer,
question_rate) values ('Складіть запит, що виводить імена всіх підлеглих
King.', 3, 'select ename from emp where mgr = (select empno from emp where
ename = ''KING'')', 3);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Складіть запит, що виводить список співробітників
компанії в одному відділі зі Smith.', 3, 'select ename from emp where deptno
= (select deptno from emp where ename = ''SMITH'')', 3);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Складіть запит, що виводить імена співробітників,
найнятих після Smith в його відділ.', 3, 'select ename from emp where
hiredate > (select hiredate from emp where ename = ''SMITH'' ) AND deptno =
(select deptno from emp where ename = ''SMITH'')', 3);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Складіть запит, що виводить імена співробітників
найнятих після службовців відділу №30', 3, 'select ename from emp where
hiredate > all (select hiredate from emp where deptno = 30)', 3);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Складіть запит, що виводить прізвище, номер
підрозділу і зарплату працівників які працюють у відділах, де деяким
співробітником виплачують премію.', 3, 'select ename, deptno, sal from emp
where deptno in (select deptno from emp where comm is not null)', 3);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Складіть запит, що виводить інформацію про
працівників, зарплата яких більше зарплати будь-якого з клерків.', 3, 'select
ename, sal from emp where sal > any (select sal from emp where job =
''CLERK'')', 3);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Складіть запит, що виводить ім''я і зарплату для всіх
```

```
співробітників з Dallas', 3, 'select ename, sal from emp where deptno =
(select deptno from dept where loc = ''DALLAS'')', 3);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Складіть запит, який виведе імена, номер відділу і
зарплату співробітників з відділу №10', 3, 'select ename, deptno, sal from
(select * from emp where deptno = 10) as e', 3);
insert into questions (question_body, category_id, question_answer,
question_rate) values ('Складіть запит, який виведе імена, номер відділу і
зарплату співробітників з відділу №10, у яких зарплата більше $2500.', 3,
'select ename, deptno, sal from (select * from emp where deptno = 10) as e
where sal > 2500', 3);
```

Скрипт заповнення стренувальної схеми

```
drop table Emp;
drop table Dept;

CREATE TABLE Dept(
   deptno integer NOT NULL UNIQUE,
   dname VARCHAR(14) NOT NULL,
   loc VARCHAR(13),
primary key (deptno)
);

CREATE TABLE Emp(
  empno integer NOT NULL UNIQUE,
  ename VARCHAR(20),
  job VARCHAR(9),
  mgr integer,
  hiredate DATE,
  sal decimal(7, 2),
  comm decimal(7, 2),
  deptno integer,
PRIMARY KEY (empno)

);


ALTER TABLE Emp ADD FOREIGN KEY (mgr) REFERENCES Emp (empno) ON update
restrict on delete restrict;
ALTER TABLE Emp ADD FOREIGN KEY (deptno) REFERENCES Dept (deptno) ON update
restrict on delete restrict;


commit;

INSERT INTO Dept  VALUES (10,'ACCOUNTING','NEW_YORK');
INSERT INTO Dept VALUES (20,'RESEARCH','DALLAS');
INSERT INTO Dept VALUES (30,'SALES','CHICAGO');
INSERT INTO Dept VALUES (40,'OPERATIONS','BOSTON');
INSERT INTO Dept VALUES (50,'RESEARCH2','HONKONG');
INSERT INTO Dept VALUES (60,'SALES2','HONKONG');
INSERT INTO Dept VALUES (100,'SALES3','NEW_YORK');


INSERT INTO Emp VALUES (7839,'KING','PRESIDENT',null, TO_DATE('2011-11-17',
'YYYY-MM-DD'),5000,null,10);
INSERT INTO Emp VALUES (7698,'BLAKE','MANAGER',7839,TO_DATE('2011-05-01',
'YYYY-MM-DD'), 2850,null,30);
INSERT INTO Emp VALUES (7782,'CLARK','MANAGER',7839,TO_DATE('2011-06-09',
'YYYY-MM-DD'), 1500,null,10);
INSERT INTO Emp VALUES (7566,'JONES','MANAGER',7839,TO_DATE('2011-04-02',
'YYYY-MM-DD'), 2975,null,20);
```

```
INSERT INTO Emp VALUES (7654,'MARTIN','SALESMAN',7698,TO_DATE('2011-09-28',
'YYYY-MM-DD'), 1250,1400,30);
INSERT INTO Emp VALUES (7499,'ALLEN','SALESMAN',7698,TO_DATE('2011-02-20',
'YYYY-MM-DD'), 1600,300,30);
INSERT INTO Emp VALUES (7844,'TURNER','SALESMAN',7698,TO_DATE('2011-09-08',
'YYYY-MM-DD'), 1500,0,30);
INSERT INTO Emp VALUES (7900,'JAMES','CLERK',7698,TO_DATE('2011-12-03',
'YYYY-MM-DD'), 950,null,30);
INSERT INTO Emp VALUES (7521,'WARD','SALESMAN',7698,TO_DATE('2011-02-22',
'YYYY-MM-DD'), 1250,500,30);
INSERT INTO Emp VALUES (7902,'FORD','ANALYST',7566,TO_DATE('2011-12-03',
'YYYY-MM-DD'), 3000,null,20);
INSERT INTO Emp VALUES (7369,'SMITH','CLERK',7902,TO_DATE('2010-12-17',
'YYYY-MM-DD'), 800,null,20);
INSERT INTO Emp VALUES (7788,'SCOTT','ANALYST',7566,TO_DATE('2012-12-09',
'YYYY-MM-DD'), 3000,null,20);
INSERT INTO Emp VALUES (7876,'ADAMS','CLERK',7788,TO_DATE('2013-01-12',
'YYYY-MM-DD'), 1100,null,20);
INSERT INTO Emp VALUES (7934,'MILLER','CLERK',7782,TO_DATE('2012-01-23',
'YYYY-MM-DD'), 1300,null,10);
INSERT INTO Emp VALUES (8000,'JACKIE CHAN','SALESMAN',7839,TO_DATE('2011-09-
28', 'YYYY-MM-DD'), 2250,1700,60);
INSERT INTO Emp VALUES (8001,'JET LI','SALESMAN',8000,TO_DATE('2011-02-20',
'YYYY-MM-DD'), 2600,600,60);
INSERT INTO Emp VALUES (8002,'BRUCE LEE','SALESMAN',8000,TO_DATE('2011-09-
08', 'YYYY-MM-DD'), 2500,null,60);
INSERT INTO Emp VALUES (8003,'DR NO','ANALYST', 7839, TO_DATE('2011-09-11',
'YYYY-MM-DD'), 2500,null,null);
commit;
```

ДОДАТОК Б. ЛІСТИНГ ПРОГРАМНОГО КОДУ

У даному додатку наведено програмний код додатку.

site.sqltrainer.app.Main

```
package site.sqltrainer.app;

import org.eclipse.jetty.server.Server;
import org.eclipse.jetty.util.thread.ThreadPool;
import site.sqltrainer.controllers.*;
import spark.Spark;
import spark.embeddedserver.EmbeddedServers;
import spark.embeddedserver.jetty.EmbeddedJettyFactory;
import spark.embeddedserver.jetty.JettyServerFactory;

import static spark.Spark.*;

public class Main {
    private static final LoginController LOGIN_CONTROLLER = new
LoginController();
    private static final RegistrationController REGISTRATION_CONTROLLER = new
RegistrationController();
    private static final MainPageController MAIN_PAGE_CONTROLLER = new
MainPageController();
    private static final CabinetController CABINET_CONTROLLER = new
CabinetController();
    private static final TaskController TASK_CONTROLLER = new
TaskController();

    public static void main(String[] args) {
        Spark.exception(Exception.class, (exception, request, response) -> {
            exception.printStackTrace();
        });
        System.out.println("started");
        port(50300);
        CustomJettyServerFactory customJettyServerFactory = new
CustomJettyServerFactory();
        EmbeddedServers.add(
                EmbeddedServers.Identifiers.JETTY,
                new EmbeddedJettyFactory(customJettyServerFactory));
        staticFileLocation("/public");

        //mainPage
        get("/mainPage", MAIN_PAGE_CONTROLLER.getMainPage);
        get("/", MAIN_PAGE_CONTROLLER.getMainPage);

        //login
        get("/login", LOGIN_CONTROLLER.getLoginPage);
        post("/login", LOGIN_CONTROLLER.tryToLogin);
        get("/logout", LOGIN_CONTROLLER.logout);

        //registration
        get("/register", REGISTRATION_CONTROLLER.getRegistrationPage);
        post("/register", REGISTRATION_CONTROLLER.registrationProcess);

        //cabinet
        get("/cabinet", CABINET_CONTROLLER.getCabinetPage);
        get("/categories", CABINET_CONTROLLER.getCategoriesPage);
        get("/schema", CABINET_CONTROLLER.getSchemaPage);
        get("/jsonData", CABINET_CONTROLLER.getJsonData);

        //tasks
```

```
        get("/viewQuestion", TASK_CONTROLLER.viewQuestion);
        get("/viewCategory", TASK_CONTROLLER.viewCategory);
        post("/postAnswer", TASK_CONTROLLER.postAnswer);

        //if not found page
        get("/*", (((request, response) -> {
            if (response.status() != 200) response.redirect("/mainPage");
            return null;
        })));
    }

    static class CustomJettyServerFactory implements JettyServerFactory {
        @Override
        public Server create(int maxThreads, int minThreads, int
threadTimeoutMillis) {
            Server server = new Server();

server.setAttribute("org.eclipse.jetty.server.Request.maxFormContentSize",
1500000);
            return server;
        }

        @Override
        public Server create(ThreadPool threadPool) {
            return null;
        }
    }
}
```

site.sqltrainer.controllers.Controller

```
package site.sqltrainer.controllers;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import site.sqltrainer.model.User;
import spark.Request;
import spark.Response;

import java.util.HashMap;

/**
 * Class-parent for controllers.
 */
public abstract class Controller {

    private static final Logger LOGGER =
LoggerFactory.getLogger(Controller.class);

    protected void makeHeader(Request request, HashMap<String, Object> model)
{
        if (request.session().attribute("user") != null) {
            model.put("cabinet", "Особистий кабінет");
            model.put("cabinetHref", "/cabinet");
            model.put("logoutText", "Вийти");
            model.put("logout", "/logout");
        } else {
            model.put("cabinet", "Увійти");
            model.put("cabinetHref", "/login");
        }
    }

    protected boolean isAlreadyLogedIn(Request request, Response response) {
```

```
        return request.session().attribute("user") != null;
    }

    protected int parsePageNumber(Request request) {
        int curPage = 1;
        try {
            String page = request.queryMap("page").value();
            if (page != null) {
                curPage = Integer.parseInt(page);
            }

            if (curPage <= 0) {
                curPage = 1;
            }
        } catch (Exception e) {
            LOGGER.error("Error parsePageNumber ", e);
        }
        return curPage;
    }

    protected void makePagination(Request request, String linkForPage, int
totalCount, HashMap<String, Object> model, int perPage) {
        int curPage = parsePageNumber(request);
        int pageCount = (int) Math.ceil((double) totalCount / perPage);
        HashMap<Integer, String> links = new HashMap<>();
        if (totalCount == 0) {
            links.put(1, linkForPage + 1);
        } else if (pageCount == 1) {
            links.put(1, linkForPage + 1);
        } else if (pageCount == 2) {
            links.put(1, linkForPage + 1);
            links.put(2, linkForPage + 2);
        } else if (curPage == 1) {
            links.put(1, linkForPage + 1);
            links.put(2, linkForPage + 2);
            links.put(3, linkForPage + 3);
        } else if (curPage == pageCount) {
            links.put((pageCount - 2), linkForPage + (pageCount - 2));
            links.put((pageCount - 1), linkForPage + (pageCount - 1));
            links.put(pageCount, linkForPage + pageCount);
        } else {
            links.put(curPage + 1, linkForPage + (curPage + 1));
            links.put(curPage, linkForPage + curPage);
            links.put(curPage - 1, linkForPage + (curPage - 1));
        }
        model.put("links", links);
        model.put("curPage", curPage);
    }

}
```

site.sqltrainer.controllers.CabinetController

```
package site.sqltrainer.controllers;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import site.sqltrainer.dao.QuestionDAOImpl;
import site.sqltrainer.dao.UserDAOImpl;
import site.sqltrainer.model.QuestionCategory;
import site.sqltrainer.model.User;
import spark.ModelAndView;
import spark.Route;
import spark.template.thymeleaf.ThymeleafTemplateEngine;
```

```
import java.util.HashMap;
import java.util.List;

/**
 * This is controller for user cabinet.
 */
public class CabinetController extends Controller {
    private final QuestionDAOImpl questionDAO = new QuestionDAOImpl();
    private final UserDAOImpl userDAO = new UserDAOImpl();
    private static final int PER_PAGE = 6;
    private static final Logger LOGGER =
LoggerFactory.getLogger(CabinetController.class);

    public final Route getCabinetPage = ((request, response) -> {
        HashMap<String, Object> model = new HashMap<>();
        User user = request.session().attribute("user");
        if (user == null) {
            model.put("message", "Ви не авторизовані");
            response.redirect("/mainPage");
            return "";
        }
        String userCategory = userDAO.getUserCategory(user.getId());
        model.put("user", user);
        model.put("userCategory", userCategory);
        return new ThymeleafTemplateEngine().render(new ModelAndView(model,
"account"));
    });

    public final Route getCategoriesPage = ((request, response) -> {
        HashMap<String, Object> model = new HashMap<>();
        User user = request.session().attribute("user");
        if (user == null) {
            model.put("message", "Ви не авторизовані");
            response.redirect("/mainPage");
            return "";
        }

        int start = (parsePageNumber(request) - 1) * PER_PAGE;
        Object[] objects =
questionDAO.getQuestionCategoriesForPagination(PER_PAGE, start);
        int categoriesCount = (int) objects[0];
        makePagination(request, "/viewCategory?id=", categoriesCount, model,
PER_PAGE);

        List<QuestionCategory> questionCategories = (List<QuestionCategory>)
objects[1];
        model.put("questionCategories", questionCategories);

        model.put("user", user);
        return new ThymeleafTemplateEngine().render(new ModelAndView(model,
"categories"));
    });

    public final Route getSchemaPage = ((request, response) -> {
        HashMap<String, Object> model = new HashMap<>();
        User user = request.session().attribute("user");
        if (user == null) {
            model.put("message", "Ви не авторизовані");
            response.redirect("/mainPage");
            return "";
        }
        model.put("user", user);
        return new ThymeleafTemplateEngine().render(new ModelAndView(model,
```

```
"schema"));
    });

    public final Route getJsonData = ((request, response) -> {
        HashMap<String, Object> model = new HashMap<>();
        User user = request.session().attribute("user");
        if (user == null) {
            model.put("message", "Ви не авторизовані");
            response.redirect("/mainPage");
            return "";
        }
        String userAnswerData = userDAO.getUserAnswersData(user.getId());
        if (userAnswerData == null || "".equals(userAnswerData)){
            userAnswerData = "[]";
        }
        return userAnswerData;
    });

}
```

site.sqltrainer.controllers.LoginController

```
package site.sqltrainer.controllers;

import site.sqltrainer.dao.UserDAOImpl;
import site.sqltrainer.model.User;
import spark.ModelAndView;
import spark.Route;
import spark.template.thymeleaf.ThymeleafTemplateEngine;

import java.util.HashMap;

/**
 * This is controller for user authorization.
 */
public class LoginController extends Controller {
    private final UserDAOImpl userDAO = new UserDAOImpl();

    public final Route getLoginPage = ((request, response) -> {
        if (isAlreadyLogedIn(request, response)) {
            response.redirect("/mainPage");
            return null;
        }
        HashMap<String, Object> model = new HashMap<>();
        makeHeader(request, model);
        model.put("user", new User());
        model.put("message", " ");
        return new ThymeleafTemplateEngine().render(
                new ModelAndView(model, "login"));
    });

    public final Route tryToLogin = ((request, response) -> {
        String login = request.queryParams("login");
        String password = request.queryParams("password");
        User user = userDAO.getUserByLoginAndPassword(login, password);
        if (user == null) {
            HashMap<String, Object> model = new HashMap<>();
            model.put("message", "Невірний логін чи пароль");
            model.put("user", new User(login, password));
            ModelAndView samePage = new ModelAndView(model, "/login");
            return new ThymeleafTemplateEngine().render(samePage);
        } else {
            request.session().attribute("user", user);
```

```
            response.redirect("/cabinet");
            return null;
        }
    });

    public Route logout = ((request, response) -> {
        if (request.session().attribute("user") != null) {
            request.session().removeAttribute("user");
            request.session().invalidate();
        }
        response.redirect("/mainPage");
        return null;
    });
}
```

site.sqltrainer.controllers.MainPageController

```
package site.sqltrainer.controllers;

import spark.ModelAndView;
import spark.Route;
import spark.template.thymeleaf.ThymeleafTemplateEngine;

import java.util.HashMap;

/**
 * This is controller for main page.
 */
public class MainPageController extends Controller {

    public final Route getMainPage = ((request, response) -> {
        HashMap<String, Object> model = new HashMap<>();
        makeHeader(request, model);
        ModelAndView mainPage = new ModelAndView(model, "index");
        return new ThymeleafTemplateEngine().render(mainPage);
    });

}
```

site.sqltrainer.controllers.RegistrationController

```
package site.sqltrainer.controllers;

import site.sqltrainer.dao.UserDAOImpl;
import site.sqltrainer.model.User;
import site.sqltrainer.services.PasswordProtectorService;
import spark.ModelAndView;
import spark.Route;
import spark.template.thymeleaf.ThymeleafTemplateEngine;

import java.util.HashMap;

/**
 * This is controller for user registration.
 */
public class RegistrationController extends Controller {
    private final UserDAOImpl userDAO = new UserDAOImpl();

    public final Route getRegistrationPage = ((request, response) -> {
        if (isAlreadyLogedIn(request, response)) {
            response.redirect("/mainPage");
            return null;
```

```
        }
        HashMap<String, Object> model = new HashMap<>();
        makeHeader(request, model);
        model.put("user", new User());
        model.put("message", " ");
        return new ThymeleafTemplateEngine().render(
                new ModelAndView(model, "registration"));
    });

    public final Route registrationProcess = ((request, response) -> {
        String login = request.queryParams("login");
        String rawPassword = request.queryParams("password");
        String name = request.queryParams("name");
        String message = null;
        User user = null;
        if (rawPassword.length() < 8) {
            message = "Пароль не може бути менше 8 символів";
        }
        if (userDAO.checkUserByLogin(login)) {
            message = "Такий логін вже використовується";
        }
        HashMap<String, Object> model = new HashMap<>();
        String password = PasswordProtectorService.hashPassword(rawPassword);
        if (message == null) {
            user = userDAO.addNewUser(name, password, login);
        }
        if (user == null && message == null) {
            message = "Щось пішло не так";
        }
        if (message != null) {
            model.put("message", message);
            model.put("user", new User(name, login, rawPassword));
            return new ThymeleafTemplateEngine().render(new
ModelAndView(model, "registration"));
        } else {
            request.session().attribute("user", user);
            makeHeader(request, model);
            response.redirect("/mainPage");
            return null;
        }
    });
}
```

site.sqltrainer.controllers.TaskController

```
package site.sqltrainer.controllers;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import site.sqltrainer.dao.AnswerDAOImpl;
import site.sqltrainer.dao.QuestionDAOImpl;
import site.sqltrainer.model.Question;
import site.sqltrainer.model.QuestionCategory;
import site.sqltrainer.model.User;
import site.sqltrainer.services.QueryService;
import spark.ModelAndView;
import spark.Request;
import spark.Response;
import spark.Route;
import spark.template.thymeleaf.ThymeleafTemplateEngine;

import java.util.HashMap;
import java.util.List;
```

```
public class TaskController extends Controller {
    private final AnswerDAOImpl answerDAO = new AnswerDAOImpl();
    private final QuestionDAOImpl questionDAO = new QuestionDAOImpl();
    private static final Logger LOGGER =
LoggerFactory.getLogger(TaskController.class);
    private static final int PER_PAGE = 6;

    public final Route viewQuestion = ((request, response) -> {
        HashMap<String, Object> model = new HashMap<>();
        User user = request.session().attribute("user");
        if (user == null) {
            model.put("message", "Ви не авторизовані");
            response.redirect("/mainPage");
            return "";
        }
        int id = parseId(request, response);
        Question question = questionDAO.getQuestion(id);

        model.put("message", " ");
        prepareNavigationForTask(id, model, question.getCategoryId());
        model.put("user", user);
        model.put("question", question);
        model.put("user_answer", " ");
        request.session().attribute("question", question);
        return new ThymeleafTemplateEngine().render(new ModelAndView(model,
"task"));
    });

    /*
    Route for view category page with tasks list.
    Contains: QuestionCategory, Questions
     */
    public final Route viewCategory = ((request, response) -> {
        HashMap<String, Object> model = new HashMap<>();
        User user = request.session().attribute("user");
        if (user == null) {
            model.put("message", "Ви не авторизовані");
            response.redirect("/mainPage");
            return "";
        }

        int categoryId = parseId(request, response);
        int start = (parsePageNumber(request) - 1) * PER_PAGE;
        Object[] objects = questionDAO.getQuestionsForPagination(PER_PAGE,
start, categoryId, user.getId());
        int questionsCount = (int) objects[0];
        makePagination(request, "/viewCategory?id=" + categoryId + "&page=",
questionsCount, model, PER_PAGE);

        List<Question> questions = (List<Question>) objects[1];
        QuestionCategory questionCategory =
questionDAO.getQuestionCategory(categoryId);
        List<Integer> answeredQuestions = (List<Integer>) objects[2];

        model.put("category", questionCategory);
        model.put("questionsToWork", questions);
        model.put("answeredQuestions", answeredQuestions);
        model.put("user", user);
        return new ThymeleafTemplateEngine().render(new ModelAndView(model,
"tasks"));
    });

    public final Route postAnswer = ((request, response) -> {
```

```
        HashMap<String, Object> model = new HashMap<>();
        User user = request.session().attribute("user");
        if (user == null) {
            model.put("message", "Ви не авторизовані");
            response.redirect("/mainPage");
            return "";
        }
        String query = request.queryParams("query_body");
        Question question = request.session().attribute("question");
        String parsedQuery = QueryService.parseQuery(query);
        prepareNavigationForTask(question.getId(), model,
question.getCategoryId());
        model.put("user_answer", query);
        if (parsedQuery == null) {
            model.put("message", "Виконувати дозволено лише Select'и");
        } else {
            if (answerDAO.isAnswerExistByUserAndQuestion(user.getId(),
question.getId())) {
                model.put("message", "Ви вже виконали це завдання, спробуйте
наступне.");
            } else {
                boolean isAnswerCorrect =
answerDAO.isUserAnswerCorrect(parsedQuery, question.getAnswer());
                if (isAnswerCorrect) {
                    model.put("message", "Правильно, перейдіть до іншого
питання.");
                    answerDAO.addNewAnswer(user.getId(), question.getId(),
question.getQuestionRate());
                } else {
                    model.put("message", "Будь ласка, спробуйте ще раз.");
                }
            }
        }
        model.put("user", user);
        model.put("question", question);
        return new ThymeleafTemplateEngine().render(new ModelAndView(model,
"task"));
    });

    private int parseId(Request request, Response response) {
        int id = 1;
        try {
            id = Integer.parseInt(request.queryMap("id").value());
        } catch (Exception e) {
            LOGGER.error("Error on parsing id", e);
        }
        if (id == 0) {
            response.redirect("/mainPage");
        }
        return id;
    }

    private void prepareNavigationForTask(int id, HashMap<String, Object>
model, int questionCategory) {
        int nextQuestionId = id + 1;
        int prevQuestionId = id - 1;
        int questionCount = QuestionDAOImpl.getQuestionCount();
        String nextQuestionLink = "/viewQuestion?id=" + nextQuestionId;
        String prevQuestionLink = "/viewQuestion?id=" + prevQuestionId;
        if (prevQuestionId <= 0) {
            prevQuestionLink = "#";
        }
        model.put("prevQuestionLink", prevQuestionLink);
        model.put("nextQuestionLink", nextQuestionLink);
```

```
    }
}
```

site.sqltrainer.dao.Queries

```
package site.sqltrainer.dao;

public class Queries {
    //user
    public static final String GET_USER_BY_ID = "SELECT * from users WHERE
user_id = ?";
    public static final String CHECK_USER_BY_LOGIN = "SELECT count(*) from
users WHERE login = ?";
    public static final String GET_USER_BY_LOGIN_AND_PASSWORD = "SELECT *
from users WHERE login = ?";
    public static final String ADD_NEW_USER = "INSERT INTO users(name, login,
password, reg_date) VALUES (?, ?, ?, ?)";
    public static final String UPDATE_USER_RATE = "update users set user_rate
= (user_rate + ?) where user_id = ?";
    public static final String GET_USER_CATEGORY = "select uc.name from users
u join user_categories uc on u.user_rate between uc.min_rate and uc.max_rate
and u.user_id = ?";

    //question
    public static final String GET_QUESTION_BY_ID = "select * from questions
where question_id = ?";
    public static final String GET_QUESTION_CATEGORY_BY_ID = "select * from
question_categories where category_id = ?";
    public static final String GET_QUESTION_CATEGORIES_COUNT = "select
count(*) from question_categories";
    public static final String GET_QUESTION_CATEGORIES_FOR_PAGINATION =
"select * from question_categories order by 1 LIMIT ? OFFSET ?";
    public static final String GET_QUESTION_COUNT_BY_CATEGORY = "select
count(*) from questions where category_id = ?";
    public static final String GET_QUESTION_COUNT = "select count(*) from
questions";
    public static final String GET_QUESTION_FOR_PAGINATION_BY_CATEGORY =
"select * from questions where category_id = ? order by 1 LIMIT ? OFFSET ?";

    //answer
    public static final String CHECK_ANSWER_BY_USER_AND_QUESTION = "select
count(*) from answers where user_id = ? and question_id = ?";
    public static final String ADD_NEW_ANSWER = "insert into answers
(user_id, question_id, answer_date) values (?, ?, ?)";
    public static final String GET_ANSWERS_DATA_BY_USER =
            "select "
                    + "    array_to_json(array_agg(data_set)) "
                    + "from "
                    + "    (select "
                    + "       to_char(answer_date, 'MON YYYY') as
answer_date"
                    + "       ,  count(answer_id)                as
answer_count "
                    + "        from   answers "
                    + "        where  user_id = ?"
                    + "        group  by to_char(answer_date, 'MON YYYY') "
                    + "        order by 1 "
                    + "    ) data_set";

    public static final String GET_ANSWERED_QUESTIONS_BY_USER_AND_CATEGORY =
            "select "
                    + "    q.question_id "
                    + "from "
                    + "    answers a "
```

```
                    + "    join questions q on a.question_id = q.question_id
"
                    + "where "
                    + "    a.user_id = ? "
                    + "    and q.category_id = ?";
}
```

site.sqltrainer.dao.AnswerDAOImpl

```
package site.sqltrainer.dao;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import site.sqltrainer.dbutils.ConnectionManager;
import site.sqltrainer.model.Answer;

import java.sql.*;
import java.time.LocalDate;

public class AnswerDAOImpl {
    private static final Logger LOGGER =
LoggerFactory.getLogger(AnswerDAOImpl.class);

    public boolean isUserAnswerCorrect(String userAnswer, String answer) {
        boolean isUserAnswerCorrect = false;
        String queryToCheck = userAnswer + " except " + answer;
        try (Connection connection =
ConnectionManager.getConnectionForTask();
             Statement stmt = connection.createStatement()) {
            ResultSet resultSet = stmt.executeQuery(queryToCheck);
            if (!resultSet.next()) {
                isUserAnswerCorrect = true;
            }
        } catch (SQLException e) {
            LOGGER.error("Error isUserAnswerCorrect " + e.getMessage());
        }
        return isUserAnswerCorrect;
    }

    public void addNewAnswer(int userId, int questionId, int answerRate) {
        try (Connection connection = ConnectionManager.getSimpleConnection();
             PreparedStatement insertStatement =
connection.prepareStatement(Queries.ADD_NEW_ANSWER);
             PreparedStatement userRate =
connection.prepareStatement(Queries.UPDATE_USER_RATE)) {
            insertStatement.setInt(1, userId);
            insertStatement.setInt(2, questionId);
            insertStatement.setDate(3, Date.valueOf(LocalDate.now()));
            insertStatement.execute();

            userRate.setInt(1, answerRate);
            userRate.setInt(2, userId);
            userRate.execute();

        } catch (SQLException e) {
            LOGGER.error("Error while adding new answer ", e);
        }
    }

    public boolean isAnswerExistByUserAndQuestion(int userId, int questionId)
{
        boolean isAnswerExist = false;
        try (Connection connection = ConnectionManager.getSimpleConnection();
```

```
             PreparedStatement statement =
connection.prepareStatement(Queries.CHECK_ANSWER_BY_USER_AND_QUESTION)) {
            statement.setInt(1, userId);
            statement.setInt(2, questionId);
            ResultSet resultSet = statement.executeQuery();
            resultSet.next();
            isAnswerExist = resultSet.getInt(1) == 1;
        } catch (SQLException e) {
            LOGGER.error("Error checkAnswerByUserAndQuestion ", e);
        }
        return isAnswerExist;
    }

    private Answer parseAnswer(ResultSet resultSet) throws SQLException {
        Answer answer = new Answer();
        answer.setAnswerId(resultSet.getInt(1));
        answer.setUserId(resultSet.getInt(2));
        answer.setQuestionId(resultSet.getInt(3));
        answer.setAnswerDate(resultSet.getDate(4).toLocalDate());
        return answer;
    }
}
```

site.sqltrainer.dao.QuestionDAOImpl

```
package site.sqltrainer.dao;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import site.sqltrainer.dbutils.ConnectionManager;
import site.sqltrainer.model.Question;
import site.sqltrainer.model.QuestionCategory;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class QuestionDAOImpl {

    private static final Logger LOGGER =
LoggerFactory.getLogger(QuestionDAOImpl.class);

    public Question getQuestion(int questionId) {
        Question question = null;
        try (Connection connection = ConnectionManager.getSimpleConnection();
             PreparedStatement getQuestion =
connection.prepareStatement(Queries.GET_QUESTION_BY_ID)) {
            getQuestion.setInt(1, questionId);
            ResultSet resultSet = getQuestion.executeQuery();
            resultSet.next();
            question = parseQuestion(resultSet);

        } catch (SQLException e) {
            LOGGER.error("Error getting questions ", e);
        }

        return question;
    }

    public static int getQuestionCountByCategory(int categoryId) {
        int questionCount = 0;
        try (Connection connection = ConnectionManager.getSimpleConnection();
             PreparedStatement count =
connection.prepareStatement(Queries.GET_QUESTION_COUNT_BY_CATEGORY)) {
```

```
            count.setFetchSize(1);
            count.setInt(1, categoryId);
            ResultSet resultSet = count.executeQuery();
            resultSet.next();
            questionCount = resultSet.getInt(1);
            resultSet.close();
        } catch (SQLException e) {
            LOGGER.error("Error while counting all questions by category",
e);
        }
        return questionCount;
    }

    public static int getQuestionCount() {
        int questionCount = 0;
        try (Connection connection = ConnectionManager.getSimpleConnection();
             PreparedStatement count =
connection.prepareStatement(Queries.GET_QUESTION_COUNT)) {
            count.setFetchSize(1);
            ResultSet resultSet = count.executeQuery();
            resultSet.next();
            questionCount = resultSet.getInt(1);
            resultSet.close();
        } catch (SQLException e) {
            LOGGER.error("Error while counting all questions ", e);
        }
        return questionCount;
    }

    public Object[] getQuestionCategoriesForPagination(int perPage, int
start) {
        int categoriesCount = 0;
        List<QuestionCategory> questionCategories = new ArrayList<>();
        try (Connection connection = ConnectionManager.getSimpleConnection();
             PreparedStatement count =
connection.prepareStatement(Queries.GET_QUESTION_CATEGORIES_COUNT);
             PreparedStatement preparedStatement =
connection.prepareStatement(Queries.GET_QUESTION_CATEGORIES_FOR_PAGINATION))
{
            count.setFetchSize(1);
            ResultSet cnt = count.executeQuery();
            cnt.next();
            categoriesCount = cnt.getInt(1);
            cnt.close();
            preparedStatement.setFetchSize(perPage);
            preparedStatement.setInt(1, perPage);
            preparedStatement.setInt(2, start);
            ResultSet resultSet = preparedStatement.executeQuery();

            while (resultSet.next()) {
                QuestionCategory questionCategory =
parseQuestionCategory(resultSet);
                questionCategories.add(questionCategory);
            }
            resultSet.close();
        } catch (SQLException e) {
            LOGGER.error("Error while getQuestionCategoriesForPagination ",
e);
        }

        return new Object[]{
                categoriesCount,
                questionCategories
        };
```

```
    }

    public Object[] getQuestionsForPagination(int perPage, int start, int
categoryId, int userId) {
        int questionCount = 0;
        List<Question> questions = new ArrayList<>();
        List<Integer> answeredQuestions = new ArrayList<>();
        try (Connection connection = ConnectionManager.getSimpleConnection();
             PreparedStatement count =
connection.prepareStatement(Queries.GET_QUESTION_COUNT_BY_CATEGORY);
             PreparedStatement preparedStatement =
connection.prepareStatement(Queries.GET_QUESTION_FOR_PAGINATION_BY_CATEGORY);
             PreparedStatement answered =
connection.prepareStatement(Queries.GET_ANSWERED_QUESTIONS_BY_USER_AND_CATEGO
RY)) {
            //get questions count
            count.setFetchSize(1);
            count.setInt(1, categoryId);
            ResultSet cnt = count.executeQuery();
            cnt.next();
            questionCount = cnt.getInt(1);
            cnt.close();

            //get questions
            preparedStatement.setFetchSize(perPage);
            preparedStatement.setInt(1, categoryId);
            preparedStatement.setInt(2, perPage);
            preparedStatement.setInt(3, start);
            ResultSet resultSet = preparedStatement.executeQuery();
            while (resultSet.next()) {
                Question question = parseQuestion(resultSet);
                questions.add(question);
            }
            resultSet.close();

            //get answered questions
            answered.setInt(1, userId);
            answered.setInt(2, categoryId);
            ResultSet answeredIds = answered.executeQuery();
            while (answeredIds.next()) {
                answeredQuestions.add(answeredIds.getInt(1));
            }

        } catch (SQLException e) {
            LOGGER.error("Error while getQuestionsForPagination ", e);
        }

        return new Object[]{
                questionCount,
                questions,
                answeredQuestions
        };
    }

    public QuestionCategory getQuestionCategory(int categoryId) {
        QuestionCategory questionCategory = null;
        try (Connection connection = ConnectionManager.getSimpleConnection();
             PreparedStatement statement =
connection.prepareStatement(Queries.GET_QUESTION_CATEGORY_BY_ID)) {
            statement.setFetchSize(1);
            statement.setInt(1, categoryId);
            ResultSet resultSet = statement.executeQuery();
            resultSet.next();
            questionCategory = parseQuestionCategory(resultSet);
```

```
        } catch (SQLException e) {
            LOGGER.error("Error getQuestionCategory ", e);
        }
        return questionCategory;
    }

    private Question parseQuestion(ResultSet resultSet) throws SQLException {
        Question question = new Question();
        question.setId(resultSet.getInt(1));
        question.setCategoryId(resultSet.getInt(2));
        question.setBody(resultSet.getString(3));
        question.setAnswer(resultSet.getString(4));
        question.setQuestionRate(resultSet.getInt(5));
        return question;
    }

    private QuestionCategory parseQuestionCategory(ResultSet resultSet)
throws SQLException {
        QuestionCategory category = new QuestionCategory();
        category.setId(resultSet.getInt(1));
        category.setName(resultSet.getString(2));
        category.setDescription(resultSet.getString(3));
        category.setLogoPath(resultSet.getString(4));
        return category;
    }
}
```

site.sqltrainer.dao.UserDAOImpl

```
package site.sqltrainer.dao;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import site.sqltrainer.dbutils.ConnectionManager;
import site.sqltrainer.model.User;
import site.sqltrainer.services.PasswordProtectorService;

import java.sql.*;
import java.time.LocalDate;

public class UserDAOImpl {

    private static final Logger LOGGER =
LoggerFactory.getLogger(UserDAOImpl.class);

    public boolean checkUserByLogin(String login) {
        boolean isUserExist = false;
        try (Connection connection = ConnectionManager.getSimpleConnection();
             PreparedStatement statement =
connection.prepareStatement(Queries.CHECK_USER_BY_LOGIN)) {
            statement.setString(1, login);
            ResultSet resultSet = statement.executeQuery();
            resultSet.next();
            isUserExist = resultSet.getInt(1) == 1;
        } catch (SQLException e) {
            LOGGER.error("Error while checking user by login ", e);
        }
        return isUserExist;
    }

    public User getUserByLoginAndPassword(String login, String password) {
        User user = null;
        try (Connection connection = ConnectionManager.getSimpleConnection();
```

```
             PreparedStatement statement =
connection.prepareStatement(Queries.GET_USER_BY_LOGIN_AND_PASSWORD)) {
            statement.setString(1, login);
            ResultSet resultSet = statement.executeQuery();
            resultSet.next();
            user = parseUser(resultSet);
            if (!PasswordProtectorService.isPasswordsEqual(password,
user.getPassword())) {
                user = null;
            }
        } catch (SQLException e) {
            LOGGER.error("Error while selecting user by login and password ",
e);
        }
        return user;
    }

    public User addNewUser(String name, String password, String login) {
        User user = null;
        try (Connection connection = ConnectionManager.getSimpleConnection();
             PreparedStatement insertStatement =
connection.prepareStatement(Queries.ADD_NEW_USER);
             PreparedStatement selectStatement =
connection.prepareStatement(Queries.GET_USER_BY_LOGIN_AND_PASSWORD);) {
            insertStatement.setString(1, name);
            insertStatement.setString(2, login);
            insertStatement.setString(3, password);
            insertStatement.setDate(4, Date.valueOf(LocalDate.now()));
            insertStatement.execute();
            selectStatement.setString(1, login);
            ResultSet resultSet = selectStatement.executeQuery();
            resultSet.next();
            user = parseUser(resultSet);
        } catch (SQLException e) {
            LOGGER.error("Error while adding new user ", e);
        }
        return user;
    }

    public String getUserCategory(int userId) {
        String userCategory = null;
        try (Connection connection = ConnectionManager.getSimpleConnection();
             PreparedStatement statement =
connection.prepareStatement(Queries.GET_USER_CATEGORY)) {
            statement.setFetchSize(1);
            statement.setInt(1, userId);
            ResultSet resultSet = statement.executeQuery();
            resultSet.next();
            userCategory = resultSet.getString(1);
        } catch (SQLException e) {
            LOGGER.error("Error getUserCategory ", e);
        }
        return userCategory;
    }

    public String getUserAnswersData(int userId) {
        String userAnswersData = null;
        try (Connection connection = ConnectionManager.getSimpleConnection();
             PreparedStatement statement =
connection.prepareStatement(Queries.GET_ANSWERS_DATA_BY_USER)) {
            statement.setFetchSize(1);
            statement.setInt(1, userId);
            ResultSet resultSet = statement.executeQuery();
            resultSet.next();
```

```
            userAnswersData = resultSet.getString(1);
        } catch (SQLException e) {
            LOGGER.error("Error getUserAnswersData ", e);
        }
        return userAnswersData;
    }

    private User parseUser(ResultSet resultSet) throws SQLException {
        User user = new User();
        user.setId(resultSet.getInt(1));
        user.setName(resultSet.getString(2));
        user.setLogin(resultSet.getString(3));
        user.setPassword(resultSet.getString(4));
        user.setRegDate(resultSet.getDate(5).toLocalDate());
        user.setUserRate(resultSet.getInt(6));
        return user;
    }
}
```

site.sqltrainer.dbutils.ConnectionManager

```
package site.sqltrainer.dbutils;

import org.postgresql.ds.PGPoolingDataSource;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
import java.util.Properties;

/**
 * Database connection management.
 */
public class ConnectionManager {
    private static final Logger LOGGER =
LoggerFactory.getLogger(ConnectionManager.class);
    private static Properties properties = new Properties();
    public static final PGPoolingDataSource SIMPLE_DATA_SOURCE = new
PGPoolingDataSource();
    public static final PGPoolingDataSource TASK_DATA_SOURCE = new
PGPoolingDataSource();
    static int opened = 0;

    static {
        //read property from file
        //set properties to dataSource
        try {
            properties.load(new
FileInputStream("configurations/db.properties"));
            SIMPLE_DATA_SOURCE.setServerName(properties.getProperty("url"));

SIMPLE_DATA_SOURCE.setDatabaseName(properties.getProperty("database"));
            SIMPLE_DATA_SOURCE.setUser(properties.getProperty("username"));

SIMPLE_DATA_SOURCE.setPassword(properties.getProperty("password"));
            SIMPLE_DATA_SOURCE.setMaxConnections(150);
            SIMPLE_DATA_SOURCE.setInitialConnections(10);


TASK_DATA_SOURCE.setServerName(properties.getProperty("task_url"));
```

```
TASK_DATA_SOURCE.setDatabaseName(properties.getProperty("task_database"));

TASK_DATA_SOURCE.setUser(properties.getProperty("task_username"));

TASK_DATA_SOURCE.setPassword(properties.getProperty("task_password"));
            TASK_DATA_SOURCE.setMaxConnections(150);
            TASK_DATA_SOURCE.setInitialConnections(10);
        } catch (IOException e) {
            LOGGER.error("Error in reading configuration file " + e);
        }
    }

    /**
     * Method for database connection
     *
     * @return connection
     */
    public static Connection getSimpleConnection() {
        Connection conn = null;
        try {
            conn = SIMPLE_DATA_SOURCE.getConnection();
            opened++;
        } catch (Exception e) {
            LOGGER.error("Error in getting connection for " +
properties.getProperty("url"), e);
        }
        return conn;
    }

    /**
     * Method for database connection for tasks execution
     *
     * @return connection
     */
    public static Connection getConnectionForTask() {
        Connection conn = null;
        try {
            conn = TASK_DATA_SOURCE.getConnection();
            opened++;
        } catch (Exception e) {
            LOGGER.error("Error in getting connection for " +
properties.getProperty("task_url"), e);
        }
        return conn;
    }
}
```

site.sqltrainer.model.Answer

```
package site.sqltrainer.model;

import java.time.LocalDate;

public class Answer {
    private int answerId;
    private int userId;
    private int questionId;
    private LocalDate answerDate;

    public Answer() {
    }

    public int getAnswerId() {
```

```
        return answerId;
    }

    public void setAnswerId(int answerId) {
        this.answerId = answerId;
    }

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public int getQuestionId() {
        return questionId;
    }

    public void setQuestionId(int questionId) {
        this.questionId = questionId;
    }

    public LocalDate getAnswerDate() {
        return answerDate;
    }

    public void setAnswerDate(LocalDate answerDate) {
        this.answerDate = answerDate;
    }
}
```

site.sqltrainer.model.Question

```
package site.sqltrainer.model;

public class Question {
    private int id;
    private String body;
    private String answer;
    private int categoryId;
    private int questionRate;

    public Question() {
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getBody() {
        return body;
    }

    public void setBody(String body) {
        this.body = body;
    }

    public String getAnswer() {
        return answer;
```

```
    }

    public void setAnswer(String answer) {
        this.answer = answer;
    }

    public int getCategoryId() {
        return categoryId;
    }

    public void setCategoryId(int categoryId) {
        this.categoryId = categoryId;
    }

    public int getQuestionRate() {
        return questionRate;
    }

    public void setQuestionRate(int questionRate) {
        this.questionRate = questionRate;
    }
}
```

site.sqltrainer.model.QuestionCategory

```
package site.sqltrainer.model;

public class QuestionCategory {

    private int id;
    private String name;
    private String description;
    private String logoPath;

    public QuestionCategory() {
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getLogoPath() {
        return logoPath;
```

```
    }

    public void setLogoPath(String logoPath) {
        this.logoPath = logoPath;
    }
}
```

site.sqltrainer.model.User

```
package site.sqltrainer.model;

import java.time.LocalDate;

public class User {
    private int id;
    private String name;
    private String login;
    private String password;
    private LocalDate regDate;
    private int userRate;

    public User() {
    }

    public User(String login, String password) {
        this.login = login;
        this.password = password;
    }

    public User(String name, String login, String password) {
        this.name = name;
        this.login = login;
        this.password = password;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getLogin() {
        return login;
    }

    public void setLogin(String login) {
        this.login = login;
    }

    public String getPassword() {
        return password;
    }
```

```
    public void setPassword(String password) {
        this.password = password;
    }

    public LocalDate getRegDate() {
        return regDate;
    }

    public void setRegDate(LocalDate regDate) {
        this.regDate = regDate;
    }

    public int getUserRate() {
        return userRate;
    }

    public void setUserRate(int userRate) {
        this.userRate = userRate;
    }

}
```

site.sqltrainer.services.PasswordProtectorService

```
package site.sqltrainer.services;

import at.favre.lib.crypto.bcrypt.BCrypt;

/**
 * Password hashing and checking it with the original
 */
public class PasswordProtectorService {
    /**

    public static String hashPassword(String password) {
        String bcryptHashString = BCrypt.withDefaults().
                hashToString(13, password.toCharArray());
        return bcryptHashString;
    }

    public static boolean isPasswordsEqual(String rawPassword, String
protectedPassword) {
        BCrypt.Result result =
BCrypt.verifyer().verify(rawPassword.toCharArray(), protectedPassword);
        return result.verified;
    }
}
```

site.sqltrainer.services.QueryService

```
package site.sqltrainer.services;

public class QueryService {

    private static final String[] RESTRICTED_OPERATIONS = {"INSERT",
"UPDATE", "DELETE", "TRUNCATE", "DROP", "ALTER", "CREATE", "GRANT"};

    public static String parseQuery(String query) {
        if (query.trim().isEmpty()) {
            return null;
        } else {
```

```
            String newQuery = query.trim().replace(';', ' ');
            for (String restrictedOperation : RESTRICTED_OPERATIONS) {
                if (newQuery.toUpperCase().contains(restrictedOperation)) {
                    return null;
                }
            }
            return newQuery;
        }
    }
}
```

histogram.js

```
// set the dimensions of the canvas
var margin = {
      top: 20,
      right: 20,
      bottom: 70,
      left: 40
   },
   width = 800 - margin.left - margin.right,
   height = 400 - margin.top - margin.bottom;

// set the ranges
var x = d3.scale.ordinal().rangeRoundBands([0, width], .05);
var y = d3.scale.linear().range([height, 0]);
// define the axis
var xAxis = d3.svg.axis()
   .scale(x)
   .orient("bottom")
var yAxis = d3.svg.axis()
   .scale(y)
   .orient("left")
   .ticks(10);

// add the SVG element
var svg = d3.select("#chart").append("svg")
   .attr("width", width + margin.left + margin.right)
   .attr("height", height + margin.top + margin.bottom)
// .attr("viewBox", '0 0 300 600')
    .call(responsivefy)
   .append("g")
   .attr("transform",
      "translate(" + margin.left + "," + margin.top + ")");

// load the data
d3.json("/jsonData", function (error, data) {

   data.forEach(function (d) {
      d.answer_date = d.answer_date;
      d.answer_count = +d.answer_count;
   });
   // scale the range of the data
   x.domain(data.map(function (d) {
      return d.answer_date;
   }));
   y.domain([0, d3.max(data, function (d) {
      return d.answer_count;
   })]);
   // add axis
   svg.append("g")
      .attr("class", "x axis")
      .attr("transform", "translate(0," + height + ")")
```

```
      .call(xAxis)
      .selectAll("text")
      .style("text-anchor", "end")
      .attr("dx", "-.8em")
      .attr("dy", "-.55em")
      .attr("transform", "rotate(-90)");
   svg.append("g")
      .attr("class", "y axis")
      .call(yAxis)
      .append("text")
      .attr("transform", "rotate(-90)")
      .attr("y", 5)
      .attr("dy", ".71em")
      .style("text-anchor", "end")
      .text("answer_count");
   // Add bar chart
   svg.selectAll("bar")
      .data(data)
      .enter().append("rect")
      .attr("class", "bar")
      .attr("x", function (d) {
         return x(d.answer_date);
      })
      .attr("width", x.rangeBand())
      .attr("y", function (d) {
         return y(d.answer_count);
      })
      .attr("height", function (d) {
         return height - y(d.answer_count);
      });

});

function responsivefy(svg) {
    // get container + svg aspect ratio
    var container = d3.select(svg.node().parentNode),
        width = parseInt(svg.style("width")),
        height = parseInt(svg.style("height")),
        aspect = width / height;

    svg.attr("viewBox", "0 0 " + width + " " + height)
        .attr("perserveAspectRatio", "xMinYMid")
        .call(resize);

    d3.select(window).on("resize." + container.attr("id"), resize);

    // get width of container and resize svg to fit it
    function resize() {
        var targetWidth = parseInt(container.style("width"));
        if(targetWidth > 800){
            targetWidth= 800;
        }
        svg.attr("width", targetWidth);
        svg.attr("height", Math.round(targetWidth / aspect));
    }
}
```

site.sqltrainer.controllers.TestRun

```
package site.sqltrainer.controllers;

import org.apache.http.Consts;
import org.apache.http.NameValuePair;
import org.apache.http.client.entity.UrlEncodedFormEntity;
```

```
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.message.BasicNameValuePair;
import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
import spark.Spark;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import static org.junit.jupiter.api.Assertions.assertEquals;

public class TestRun {

    @BeforeAll
    public static void setup() {
        SparkServer.main(null);
    }

    @AfterAll
    public static void tearDown() throws Exception {
        Thread.sleep(1000);
        Spark.stop();
    }

    @Test
    public void testGetMainPage() throws IOException {
        SparkServer.getMainPage();
        CloseableHttpClient httpClient = HttpClients.custom().build();
        HttpGet httpGet = new HttpGet("http://localhost:8888/mainPage");
        CloseableHttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        assertEquals(200, statusCode);
    }

    @Test
    public void testGetLoginPage() throws IOException {
        SparkServer.getLoginPage();
        CloseableHttpClient httpClient = HttpClients.custom().build();
        HttpGet httpGet = new HttpGet("http://localhost:8888/login");
        CloseableHttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        assertEquals(200, statusCode);
    }

    @Test
    public void testPostLoginPage() throws IOException {
        SparkServer.postLoginPage();
        CloseableHttpClient httpClient = HttpClients.custom().build();
        List<NameValuePair> form = new ArrayList<>();
        form.add(new BasicNameValuePair("login", "TestUser"));
        form.add(new BasicNameValuePair("password", "TestUser"));
        UrlEncodedFormEntity entity = new UrlEncodedFormEntity(form,
Consts.UTF_8);
        HttpPost httpPost = new HttpPost("http://localhost:8888/login");
        httpPost.setEntity(entity);
        CloseableHttpResponse response = httpClient.execute(httpPost);
        String location = response.getFirstHeader("Location").getValue();
        int statusCode = response.getStatusLine().getStatusCode();
```

```
        assertEquals(302, statusCode);
        assertEquals("http://localhost:8888/cabinet", location);
    }

    @Test
    public void testGetRegistrationPage() throws IOException {
        SparkServer.getRegistrationPage();
        CloseableHttpClient httpClient = HttpClients.custom().build();
        HttpGet httpGet = new HttpGet("http://localhost:8888/register");
        CloseableHttpResponse response = httpClient.execute(httpGet);

        int statusCode = response.getStatusLine().getStatusCode();
        assertEquals(200, statusCode);
    }

    @Test
    public void testPostRegistrationPage() throws IOException {
        SparkServer.postRegistrationPage();
        CloseableHttpClient httpClient = HttpClients.custom().build();

        List<NameValuePair> form = new ArrayList<>();
        form.add(new BasicNameValuePair("login", "AutoTestUser"));
        form.add(new BasicNameValuePair("name", "AutoTestUser"));
        form.add(new BasicNameValuePair("password", "AutoTestUser"));
        UrlEncodedFormEntity entity = new UrlEncodedFormEntity(form,
Consts.UTF_8);
        HttpPost httpPost = new HttpPost("http://localhost:8888/register");
        httpPost.setEntity(entity);

        CloseableHttpResponse response = httpClient.execute(httpPost);
        String location = response.getFirstHeader("Location").getValue();
        int statusCode = response.getStatusLine().getStatusCode();
        assertEquals(302, statusCode);
        assertEquals("http://localhost:8888/mainPage", location);
    }

    @Test
    public void testGetCabinetPage() throws IOException {
        CloseableHttpClient httpClient = HttpClients.custom().build();
        executeLogin(httpClient);
        SparkServer.getCabinetPage();
        HttpGet httpGet = new HttpGet("http://localhost:8888/cabinet");
        CloseableHttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        assertEquals(200, statusCode);
    }

    @Test
    public void testGetCategoriesPage() throws IOException {
        CloseableHttpClient httpClient = HttpClients.custom().build();
        executeLogin(httpClient);
        SparkServer.getCategoriesPage();
        HttpGet httpGet = new HttpGet("http://localhost:8888/categories");
        CloseableHttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        assertEquals(200, statusCode);
    }

    @Test
    public void testGetJsonData() throws IOException {
        CloseableHttpClient httpClient = HttpClients.custom().build();
        executeLogin(httpClient);
        SparkServer.getJsonData();
        HttpGet httpGet = new HttpGet("http://localhost:8888/jsonData");
```

```
        CloseableHttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        assertEquals(200, statusCode);
    }

    @Test
    public void testGetSchemaPage() throws IOException {
        CloseableHttpClient httpClient = HttpClients.custom().build();
        executeLogin(httpClient);
        SparkServer.getSchemaPage();
        HttpGet httpGet = new HttpGet("http://localhost:8888/schema");
        CloseableHttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        assertEquals(200, statusCode);
    }

    @Test
    public void testViewCategory() throws IOException {
        CloseableHttpClient httpClient = HttpClients.custom().build();
        executeLogin(httpClient);
        SparkServer.viewCategory();
        HttpGet httpGet = new
HttpGet("http://localhost:8888/viewCategory?id=1");
        CloseableHttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        assertEquals(200, statusCode);
    }

    @Test
    public void testViewQuestion() throws IOException {
        CloseableHttpClient httpClient = HttpClients.custom().build();
        executeLogin(httpClient);
        SparkServer.viewQuestion();
        HttpGet httpGet = new
HttpGet("http://localhost:8888/viewQuestion?id=34");
        CloseableHttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        assertEquals(200, statusCode);
    }

    @Test
    public void testPostAnswer() throws IOException {
        CloseableHttpClient httpClient = HttpClients.custom().build();
        executeLogin(httpClient);
        SparkServer.viewQuestion();
        HttpGet httpGet = new
HttpGet("http://localhost:8888/viewQuestion?id=34");
        httpClient.execute(httpGet);
        SparkServer.postAnswer();
        List<NameValuePair> form = new ArrayList<>();
        form.add(new BasicNameValuePair("query_body", "select * from emp"));
        UrlEncodedFormEntity entity = new UrlEncodedFormEntity(form,
Consts.UTF_8);
        HttpPost httpPost = new HttpPost("http://localhost:8888/postAnswer");
        httpPost.setEntity(entity);
        CloseableHttpResponse response = httpClient.execute(httpPost);
        int statusCode = response.getStatusLine().getStatusCode();
        assertEquals(200, statusCode);
    }

    private void executeLogin(CloseableHttpClient httpClient) throws
IOException {
        SparkServer.postLoginPage();
        List<NameValuePair> form = new ArrayList<>();
```

```
        form.add(new BasicNameValuePair("login", "TestUser"));
        form.add(new BasicNameValuePair("password", "TestUser"));
        UrlEncodedFormEntity entity = new UrlEncodedFormEntity(form,
Consts.UTF_8);
        HttpPost httpPost = new HttpPost("http://localhost:8888/login");
        httpPost.setEntity(entity);
        httpClient.execute(httpPost);
    }

    public static class SparkServer {
        private static final MainPageController MAIN_PAGE_CONTROLLER = new
MainPageController();
        private static final LoginController LOGIN_CONTROLLER = new
LoginController();
        private static final RegistrationController REGISTRATION_CONTROLLER =
new RegistrationController();
        private static final CabinetController CABINET_CONTROLLER = new
CabinetController();
        private static final TaskController TASK_CONTROLLER = new
TaskController();

        public static void main(String[] args) {
            Spark.port(8888);
            Spark.threadPool(1000, 1000, 60000);
        }

        //mainPage
        public static void getMainPage() {
            Spark.get("/mainPage", MAIN_PAGE_CONTROLLER.getMainPage);
        }

        //login
        public static void getLoginPage() {
            Spark.get("/login", LOGIN_CONTROLLER.getLoginPage);
        }

        public static void postLoginPage() {
            Spark.post("/login", LOGIN_CONTROLLER.tryToLogin);
        }

        //registration
        public static void getRegistrationPage() {
            Spark.get("/register",
REGISTRATION_CONTROLLER.getRegistrationPage);
        }

        public static void postRegistrationPage() {
            Spark.post("/register",
REGISTRATION_CONTROLLER.registrationProcess);
        }

        //cabinet
        public static void getCabinetPage() {
            Spark.get("/cabinet", CABINET_CONTROLLER.getCabinetPage);
        }

        public static void getJsonData() {
            Spark.get("/jsonData", CABINET_CONTROLLER.getJsonData);
        }

        public static void getCategoriesPage() {
            Spark.get("/categories", CABINET_CONTROLLER.getCategoriesPage);
        }
```

```
        public static void getSchemaPage() {
            Spark.get("/schema", CABINET_CONTROLLER.getSchemaPage);
        }

        //tasks
        public static void viewQuestion() {
            Spark.get("/viewQuestion", TASK_CONTROLLER.viewQuestion);
        }

        public static void viewCategory() {
            Spark.get("/viewCategory", TASK_CONTROLLER.viewCategory);
        }

        public static void postAnswer() {
            Spark.post("/postAnswer", TASK_CONTROLLER.postAnswer);
        }

    }

}
```