ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка програмного забезпечення для підтримки процесів тестування в Київському регіональному центрі підвищення кваліфікації»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

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

_____ Станіслав ВОЛНЯНСЬКИЙ
(підпис)

Виконав: здобувач вищої освіти групи ПД-43

Станіслав ВОЛНЯНСЬКИЙ

Керівник: Владислав ЯСКЕВИЧ
к.т.н., доцент

Рецензент:

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Волнянському Станіславу Руслановичу

1. Тема кваліфікаційної роботи: «Розробка програмного забезпечення для підтримки процесів тестування в Київському регіональному центрі підвищення кваліфікації »
керівник кваліфікаційної роботи к.т.н., доцент Владислав ЯСКЕВИЧ,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56.

2. Строк подання кваліфікаційної роботи «02» червня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про методи розробки веб-застосунків, опис роботи веб-додатків, документація фреймворків Spring Boot та Thymeleaf.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Ознайомитися з існуючими аналогами систем тестування, оцінити їх переваги та недоліки з точки зору функціональності, зручності користування та надійності.

2. Розробити структуру бази даних для збереження тестів, питань, варіантів відповідей та результатів користувачів із можливістю багаторазового проходження тестів.
3.Реалізувати механізм збереження відповідей користувача та підрахунку результатів, з урахуванням правильності відповідей та нарахуванням балів.
4. Розробити функціонал експорту результатів тестування у форматі Excel для подальшого аналізу, зберігання або друку.
5. Розробити функціонал входу адміністратора, для редагування та створення нових тестів.
6. Розробити можливість перегляду результатів тестування користувача у вигляді неправильних відповідей та загального балу.

5. Перелік графічного матеріалу: *презентація*
1. Аналіз аналогів.
2. Вимоги до додатку.
3. Програмні засоби реалізації.
4. Загальна схема бази даних додатку.
5. Діаграма використання додатку.
6. Діаграма класів.
7. Екранні форми.
8.Апробація результатів дослідження.

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Ознайомлення з існуючими аналогами систем тестування, вивчення їх функціоналу, переваг та недоліків	25.02.-04.03.2025	
2	Аналіз отриманої інформації та визначення ключових вимог до майбутньої системи.	05.03-13.03.2025	
3	Початок проєктування структури бази даних.	15.03-21.03.2025	
4	Завершення розробки структури бази даних із можливістю багаторазового проходження тестів.	22.03-28.03.2025	
5	Узгодження структури та її тестування.	05.04-15.04.2025	
6	Початок реалізації механізму збереження відповідей користувачів та підрахунку результатів.	16.04-25.04.2025	

7	Тестування та налагодження механізму збереження відповідей та підрахунку результатів.	25.04-30.04.2025	
8	Виправлення виявлених помилок.	02.05-05.05.2025	
9	Розробка функціоналу входу адміністратора (створення, редагування тестів).	06.05-08.05.2025	
10	Розробка функціоналу експорту результатів тестування у форматі Excel для подальшого аналізу, зберігання або друку.	09.05-11.05.2025	
11	Розробка демонстраційних матеріалів	12.05-18.05.2025	
12	Попередній захист роботи	19.05-30.05.2025	

Здобувач вищої освіти _____ Станіслав ВОЛНЯНСЬКИЙ
(підпис)

Керівник
кваліфікаційної роботи _____ Владислав ЯСКЕВИЧ
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 51 стор., 1 табл., 8 рис., 20 джерел.

Мета роботи - допомога в організації та проведенні онлайн-тестування державних службовців з метою перевірки рівня їхніх знань та навичок.

Об'єкт дослідження - процес підвищення кваліфікації державних службовців із застосуванням інформаційних технологій

Предмет дослідження - застосунок пов'язаний з тестуванням та оцінюванням знань державних службовців у системі підвищення кваліфікації.

Короткий зміст роботи: У роботі проаналізовано предметну галузь підвищення кваліфікації державних службовців та освітнього процесу у сфері післядипломної освіти. Досліджено існуючі системи онлайн-тестування, їхні переваги та недоліки з точки зору функціональності, зручності використання та надійності. На основі проведеного аналізу розроблено власну концепцію веб-застосунку для тестування.

Спроєктовано архітектуру клієнтської та серверної частини проєкту. Реалізовано основні функціональні можливості: створення та редагування тестів адміністратором, проходження тестів користувачами, збереження результатів, підрахунок та аналіз результатів, а також експорт даних у форматі Excel для подальшої обробки.

Для реалізації серверної частини застосовано Java та Spring Boot, для клієнтської — HTML, CSS та JavaScript. Також налаштовано взаємодію між компонентами через контролери та репозиторії, використано базу даних для збереження тестів, відповідей і результатів. Проєкт завантажено на GitHub та проведено тестування коректності роботи системи.

КЛЮЧОВІ СЛОВА: BACK-END, FRONT-END, IDE, ВЕБДОДАТОК, RESTful API, THYMELEAF, SPRING BOOT, CSS

ЗМІСТ

ВСТУП

У сучасному освітньому середовищі цифрові технології дедалі частіше використовуються для організації навчального процесу та оцінювання знань. Одним із найефективніших засобів у цьому напрямі є онлайн-застосунки для тестування, які дозволяють автоматизувати перевірку знань, зменшити навантаження на викладачів та забезпечити об’єктивність результатів.

Вебзастосунок забезпечує централізований доступ до тестів, дозволяє поступове проходження питань, фіксацію відповідей, а також адміністрування контенту через окрему панель управління. Такий підхід створює зручні умови як для користувачів, які проходять тестування, так і для адміністраторів, які мають можливість редагувати базу питань, створювати нові тести та контролювати процес оцінювання.

Вебплатформа, дозволяє забезпечити прозоре, ефективне та безпечне проходження тестування. Гнучкість системи, можливість авторизації адміністратора та використання сучасних вебтехнологій роблять додаток актуальним рішенням для установ, які прагнуть модернізувати процес контролю знань та підвищити його ефективність у відповідності до вимог часу.

Об’єктом дослідження є процес підвищення кваліфікації державних службовців із застосуванням інформаційних технологій.

Предметом дослідження є застосунок пов’язаний з тестуванням та оцінюванням знань державних службовців у системі підвищення кваліфікації.

Метою роботи є створення веб-застосунку для проведення онлайн-тестування державних службовців. Основними завданнями системи є розробка інтуїтивного інтерфейсу для користувача з поетапним відображенням питань тесту, реалізація механізмів реєстрації та авторизації користувачів, включно з можливістю входу адміністратора.

Передбачено створення функціоналу для адміністрування тестів, включаючи додавання, редагування та видалення запитань. Також система

повинна забезпечувати адаптивне відображення на різних пристроях і підтримувати зручний перегляд статистики тестування.

Методи дослідження: Для реалізації та дослідження веб-застосунку для онлайн-тестування державних службовців було використано комплекс методів, що дозволили ефективно спроєктувати, розробити та перевірити працездатність системи. Метод моделювання дозволив створити UML-діаграми (зокрема, діаграму компонентів та структури бази даних), які відображають логіку роботи системи та взаємодію її елементів.

Це забезпечило наочне представлення внутрішньої організації додатку та спростило реалізацію його функціоналу.

У процесі розробки застосовувалися аналітичний метод, який дав змогу дослідити існуючі рішення в галузі онлайн-тестування, визначити їх переваги й недоліки, а також сформулювати вимоги до майбутньої системи. Проєктний метод був використаний для побудови архітектури клієнтської та серверної частини застосунку, розробки структури бази даних та вибору технологічного стеку, зокрема використання Spring Boot на серверній стороні та HTML/CSS/Thymeleaf для формування інтерфейсу.

Наукова новизна роботи полягає в розробці та впровадженні веб-застосунку для онлайн-тестування державних службовців, що поєднує індивідуалізований підхід до користувача, автоматизований підрахунок результатів та зручну систему адміністрування. Запропоноване рішення базується на сучасних технологіях веб-розробки (Spring Boot, Thymeleaf), що забезпечує високу продуктивність, масштабованість та безпеку системи.

Інноваційним елементом є інтеграція гнучкої системи управління тестами, яка дозволяє не лише зберігати та обробляти результати, але й здійснювати їх експорт у форматі Excel для подальшого аналізу. Також застосовано підхід поетапного проходження тестів із динамічним відображенням запитань, що підвищує зручність користування та дозволяє краще контролювати процес перевірки знань.

Усе це в сукупності формує ефективну платформу для електронного оцінювання, адаптовану до потреб державного управління.

Практична значущість полягає у можливості використання даного вебзастосунку для перевірки знань працівників, студентів або слухачів курсів без потреби в складному розгортанні чи зовнішніх сервісах. Система адаптована до використання на різних пристроях, що забезпечує широке коло застосування — як у навчальних закладах, так і в органах державної влади чи комерційних структурах.

Адміністратор може створювати та редагувати тести без знань програмування, що робить застосунок зручним і доступним для широкого кола користувачі

1 АНАЛІЗ ЗАСТОСУНКУ ДЛЯ ТЕСТУВАННЯ

1.1 Вебсайт для тестування

Актуальність: цифровізація освітніх процесів та розвиток онлайн-інструментів для оцінювання знань є невід'ємною частиною сучасного суспільства. Особливої уваги потребує впровадження ефективних засобів тестування у сфері державного управління, де контроль знань та кваліфікації працівників безпосередньо впливає на якість прийняття рішень та виконання функцій. З огляду на це, зростає попит на зручні та доступні веб-рішення, які дозволяють організувати проходження тестів без необхідності паперових носіїв або складних процедур.

Онлайн-тестування активно впроваджується у загальну, вищу, професійну та післядипломну освіту, а також у системах підвищення кваліфікації, рекрутингу, сертифікації фахівців. Завдяки розвитку веб-технологій та широкому доступу до інтернету, користувачі отримали можливість проходити тести дистанційно у зручний для себе час, що значно підвищує гнучкість освітнього процесу.[13]

Водночас із популяризацією онлайн-тестування зростають вимоги до функціональності відповідного програмного забезпечення

Основні компоненти вебсайту для тестування включають:

- Система повинна мати інтуїтивний інтерфейс для викладачів та адміністраторів, який дозволяє швидко створювати запитання, змінювати їх порядок, формувати різні варіанти тестів тощо.

- Платформи мають надавати можливість додавати запитання з одним правильним варіантом, множинним вибором, відкритою відповіддю, завдання на відповідність, вставку пропущених слів, а також інтеграцію з зображеннями, аудіо та відео для мультимедійних тестів.

- Важливо, щоб система самостійно оцінювала тести одразу після проходження, показувала кількість набраних балів, правильні/неправильні відповіді, і при потребі — надавала зворотний зв’язок.

- Результати тестування мають зберігатися у внутрішній базі даних та надаватися у вигляді таблиць або звітів для подальшого аналізу. Підтримка експорту в формати Excel, є обов’язковою вимогою для зручної роботи з даними.

- Системи повинні мати функціонал для аналізу результатів: загальна статистика, середній бал, кількість спроб, найчастіше допущені помилки, індивідуальні звіти.

- Тестова система має забезпечувати автентифікацію користувачів, захист особистих даних, контроль часу тестування, обмеження повторного проходження, а також механізми запобігання шахрайству.[4]

- Усі сторінки вебсайту оформлено з урахуванням зручності користувачів-як звичайних, так і адміністраторів.

Основні цілі вебсайту для тестування:

- Вебсайт дозволяє оцінювати рівень підготовки користувачів без участі викладача.

- Завдяки структурованим тестам користувачі можуть ефективно повторювати матеріал і закріплювати знання.

- Адміністратори мають можливість швидко створювати, змінювати й видаляти тести, питання та варіанти відповідей через спеціальну панель керування.

- Підтримка як одиночного, так і множинного вибору відповідей дозволяє створювати різні типи завдань, що відповідають методичним потребам.

- Платформа стимулює користувачів самостійно вивчати матеріал, проходити тести та контролювати власний прогрес.

Переваги вебсайту:

- Зручне створення тестів, що дає можливість людині без знань мов програмування легко створювати тести.

- База даних для зберігання результатів тестування, що дає можливість моніторингу результатів проходження тестування великої кількості користувачів

- Можливість багаторазового проходження тестування.

- Автоматичний підрахунок балів що знижує час та сили для підрахунку балів людині.

- Можливість експорту результатів в Excel дає можливість легкого аналізу та можливість друку результатів тестування.

- Вивід неправильних відповідей користувачеві по завершенню тестування, що дає можливість користувачеві побачити свої помилки та опрацювати їх

- Інтуїтивно зрозумілий інтерфейс допомагає користувачам легко зрозуміти навігацію в сайті

Недоліки:

- Потреба у стабільному інтернет-з'єднанні та сучасному обладнанні може стати перешкодою для деяких користувачів

1.2 Специфіка тестування.

Процес тестування як метод оцінювання знань і навичок має низку характерних рис, які визначають його ефективність, об’єктивність та зручність використання. Тестування широко застосовується в освіті, професійній підготовці, сертифікації, рекрутингу та інших сферах, де важливо швидко й надійно визначити рівень засвоєння матеріалу. Воно дозволяє стандартизовано оцінити велику кількість людей за короткий час із мінімальним впливом людського фактора.

Сучасні тести можуть охоплювати будь-який навчальний або професійний контекст — від перевірки мовних знань до оцінювання технічних, логічних чи аналітичних компетенцій. Вони варіюються за структурою, типами запитань та рівнем складності, що дозволяє адаптувати інструмент під різні цілі оцінювання.

Типові формати запитань:

- Вибір однієї або кількох правильних відповідей — найпоширеніший тип, що дозволяє швидко обробляти результати автоматично;

- Встановлення відповідності — використовується для перевірки логічного зв’язку між поняттями;

- Відкриті питання на коротку відповідь — дозволяють оцінити глибину розуміння теми.

Функціональні та технічні вимоги до систем тестування:

- Застосунок має забезпечувати реєстрацію та автентифікацію користувачів з розмежуванням прав доступу для звичайних користувачів і адміністраторів;

- Система повинна підтримувати збереження та обробку даних тестування користувачів, зокрема результатів проходження, обраних відповідей та балів;

- Має бути реалізована можливість створення та редагування тестів адміністраторами, включно з додаванням питань і варіантів відповідей;

- Проходження тестування користувачами повинно бути зручним, із покроковим відображенням питань та фіксацією результатів;

- Необхідно забезпечити автоматизовану оцінку результатів з урахуванням правильності відповідей і нарахуванням відповідної кількості балів;

- Повинна бути реалізована функція авторизованого входу адміністратора для керування структурою тестів, перегляду результатів та внесення змін у систему.

Організаційні та безпекові вимоги:

- Система повинна бути стабільною, захищеною від стороннього втручання, мати механізми резервного копіювання;

- Потрібна авторизація адміністратора для внесення змін у структуру тестів, список користувачів або результати;

- Важливо мати механізми захисту від несанкціонованого доступу, зокрема використання ролей, логінів і паролів;

- Для високої достовірності результатів рекомендується запобігати одночасному проходженню одного тесту з кількох пристроїв, встановлювати таймери, а також обмежувати перегляд правильних відповідей до завершення тесту.

1.3 Цільова аудиторія

Застосунок для тестування орієнтований на широке коло користувачів.

- Школи, коледжі, університети, які потребують інструментів для проведення контрольних робіт, іспитів та оцінювання знань студентів у дистанційному або очному форматі.

- Приватні або корпоративні організації, що проводять підвищення кваліфікації та сертифікацію слухачів.

- Компанії, які використовують тести для попереднього відбору кандидатів, оцінки навичок працівників або внутрішньої атестації.

- Люди, які готуються до іспитів, співбесід, сертифікацій або просто бажають перевірити власні знання.

- Фахівці, які створюють, редагують та аналізують тести, тому їм важливо мати зручні інструменти керування тестовими матеріалами та результатами.

1.4 Дослідження існуючих аналогів

На сьогоднійшій день одні з найпопулярніших застосунків є Google Forms та Moodle.

1.4.1 Google Forms

Google Forms - це веб-застосунок від Google, який дозволяє легко створювати опитування, анкети та тести. Інструмент вирішує проблему швидкого збору та аналізу даних, забезпечуючи зручний спосіб оцінювання

знань або зворотного зв’язку. Завдяки простому інтерфейсу та інтеграції з Google Sheets, користувачі можуть оперативно обробляти відповіді та візуалізувати результати. Google Forms часто використовується у сфері освіти для створення електронних тестів, опитувань та домашніх завдань, забезпечуючи базовий рівень взаємодії між викладачем та учнем.

Основні функції Google Forms:

- Можливість швидко створювати анкети, опитування, тести та реєстраційні форми.

- Підтримка різноманітних форматів, таких як тести з одним варіантом відповіді, множинний вибір, шкали оцінювання, відкриті запитання.

- Функція ключа відповідей дозволяє налаштувати автоматичне оцінювання тестів.

- Відповіді автоматично збираються та оновлюються у Google Таблицях для зручного аналізу.

- Можливість обмежити форму лише для певних користувачів або доменів.

- Форми можуть бути відкритими, приватними або з обмеженим доступом.

- Форми можна надсилати через email, вставляти на сайти або генерувати коротке посилання.

Переваги:

- Інтуїтивно зрозумілий інтерфейс, який дозволяє створити форму менш ніж за кілька хвилин.

- Швидке опрацювання відповідей завдяки інтеграції з Google Таблицями.

- Підходить як для освіти, так і для бізнесу, подій чи анкетування.

- Усі базові функції доступні безкоштовно.

Недоліки:

- Немає розгалуженої логіки оцінювання, складних типів запитань чи адаптивного тестування.

- Після проходження форми не можна реалізувати адаптивне навчання або автоматичне пояснення помилок.

- Вигляд форми майже не можна змінювати або кастомізувати під бренд чи стиль курсу.

- Не можна додавати відео або анімацію безпосередньо до форми для пояснення теорії.

- Немає ролей, журналів активності, індивідуальних кабінетів студентів.

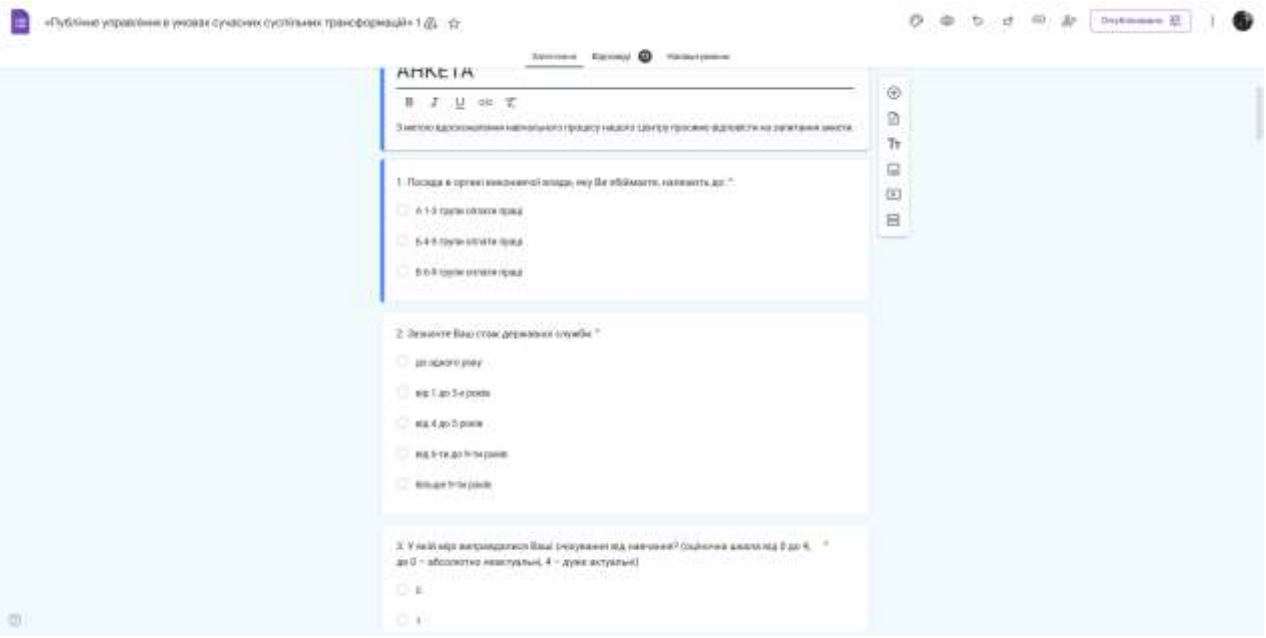


Рис. 1.1 Приклад тестування в Google Forms

1.4.2 Moodle

Moodle - це потужна платформа з відкритим кодом для управління навчанням (Learning Management System, LMS), що дозволяє створювати повноцінні онлайн-курси та ефективно організовувати освітній процес у цифровому середовищі. Вона вирішує проблему централізованої організації навчання, надаючи викладачам і адміністраторам гнучкий набір інструментів для створення уроків, тестів, форумів, завдань, оцінювання та відстеження прогресу студентів.

Moodle підтримує широкий спектр освітніх форматів — від традиційного супроводу аудиторного навчання, до змішаного (blended learning) та повністю дистанційного формату. Це дозволяє адаптувати платформу до потреб як формальної освіти (школи, університети), так і корпоративного навчання.

Основні функції Moodle:

- Створення структурованих курсів з модулями, темами, завданнями, тестами та форумами.

- Потужна система оцінювання з підтримкою різних шкал, критеріїв, журналів оцінок.

- Створення складних тестів з випадковими питаннями, таймерами, порогами проходження.

- Внутрішні комунікаційні засоби для викладачів і студентів.

- Підтримка сотень плагінів для додаткових функцій — інтеграція відео, аналітика, звіти.

- Офіційний додаток для Android та iOS дозволяє проходити курси з телефону.

- Системи відстеження активності та успішності студентів.

Переваги:

- Moodle може використовуватись як для невеликих курсів, так і для повноцінних університетських платформ.

- Можливість повного контролю, кастомізації, самостійного хостингу.

- Підтримка всіх компонентів навчального процесу — навчання, оцінювання, комунікація, аналітика.

- Доступний інтерфейс понад 100 мовами.

Недоліки:

- Незважаючи на постійні оновлення, Moodle іноді виглядає менш сучасно порівняно з новими платформами.

- Початкове встановлення та конфігурування системи може вимагати технічної підготовки або допомоги спеціаліста.

- При великій кількості користувачів або активності потрібен потужний хостинг і постійна оптимізація.

- Новачкам складно зорієнтуватися в інтерфейсі через велику кількість модулів, налаштувань та параметрів.

- Офіційний додаток не завжди стабільно працює і має обмежену функціональність.

- Для ефективного використання Moodle викладачам потрібно проходити навчання або мати певний досвід роботи з системами LMS.

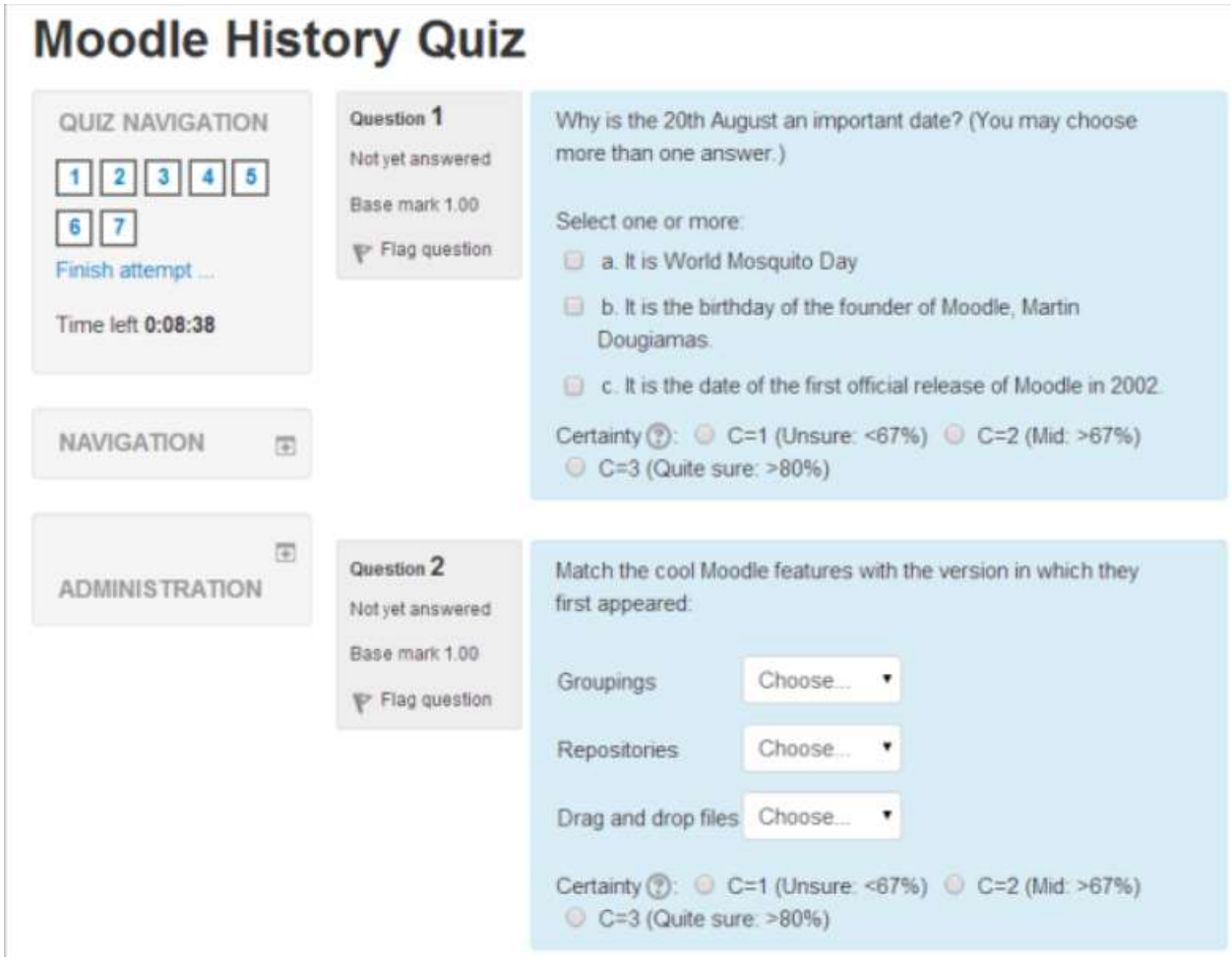


Рис.1.2 Екранні форми Moodle

В таблиці 1.1 зведено результати аналізу існуючих застосунків для тестування та їх порівняння.

Таблиця 1.1

Зведена таблиця існуючих аналогів

Функціонал	Google Forms	Moodle
Зручність створення тестів	Простий та швидкий інтерфейс для створення форм із запитаннями.	Створення тестів потребує ознайомлення з системою
База даних	Відповіді та питання зберігаються	Є вбудована об’єктно-реляційна БД

Продовження таблиці 1.1

Зведена таблиця існуючих аналогів

Додаток	Google Forms	Moodle
Можливість багаторазового проходження	Один користувач може обмежено проходити форму.	Повна підтримка багаторазових спроб із веденням історії, балів та відповідей.
Автоматичний підрахунок балів	Є базова підтримка.	Є вбудована оцінювальна система з багатьма критеріями.
Експорт результатів(Excel)	Потребує ручного експорту з Google Sheets.	Підтримується повноцінний експорт результатів в Excel
Аналіз неправильних відповідей для користувача	Відсутній.	Є зворотний зв’язок та пояснення по кожній відповіді.
Інтуїтивний інтерфейс	Інтерфейс простий і знайомий широкому колу користувачів.	Інтерфейс складний і часто перевантажений функціональністю.

1.5 Визначення вимог до застосунку

Проаналізувавши сутність застосунку, його специфіку, цільову аудиторію та аналогів, було визначено наступні вимоги до застосунку для тестування державних службовців:

- Реєстрація та автентифікація користувачів із поділом на ролі (звичайний користувач та адміністратор);

- Надійне зберігання персональних даних користувачів та результатів проходження тестування;

- Можливість створення, редагування та видалення тестів адміністратором, включаючи додавання питань, відповідей і вказання правильних варіантів;

- Зручне проходження тестування з покроковою навігацією між питаннями, та фіксацією вибраних відповідей;

- Автоматичне оцінювання результатів з нарахуванням балів відповідно до заданих критеріїв;

- Інтерфейс адміністратора для керування користувачами, тестами.

- Експорт результатів тестування у формат Excel для подальшого аналізу або архівування;

- Адаптивний інтерфейс, зручний для роботи як на ПК, так і на мобільних пристроях;

- Можливість створення тестів із питаннями, що допускають кілька правильних відповідей.

2 ЗАСОБИ РЕАЛІЗАЦІЇ

2.1 Середовище розробки

Інтегроване середовище розробки (IDE) - це комплексне програмне забезпечення, що надає програмістам різноманітні інструменти для розробки програмного коду. Інтегровані середовища розробки зазвичай включають текстовий редактор, інструменти для автоматизації збірки коду, дебагер та, часто, інтерфейс для системи контролю версій. Також вони можуть мати вбудовану підтримку систем управління базами даних, засоби автодоповнення коду, перевірку синтаксису в реальному часі, рефакторинг, емулятори, візуальні редактори інтерфейсів, інструменти для тестування та профілювання продуктивності.

IDE значно підвищують продуктивність розробника, забезпечуючи зручну навігацію по проєкту, швидкий доступ до потрібних класів, функцій або бібліотек, а також можливість миттєвого запуску або налагодження програмного коду без переходу в інші інструменти. Сучасні IDE підтримують розширення та плагіни, що дозволяє адаптувати середовище під конкретні потреби та технології.

2.1.1 Intellij IDEA

IntelliJ IDEA [15] це потужне інтегроване середовище розробки (IDE), розроблене компанією JetBrains, яке широко використовується для створення програмного забезпечення на Java та інших мовах програмування. Завдяки своїй гнучкості, розширеним можливостям аналізу коду та глибокій інтеграції з популярними фреймворками, IntelliJ IDEA є одним із найпопулярніших інструментів серед Java-розробників.

IntelliJ IDEA підтримує не лише Java, але й Kotlin, JavaScript [5], TypeScript, SQL, Python та багато інших мов. Це досягається як вбудованими засобами, так і через систему плагінів, що дозволяє налаштувати середовище відповідно до потреб користувача.

Однією з головних переваг IntelliJ IDEA є його інтелектуальна система підказок, автодоповнення та аналізу коду, що значно пришвидшує написання коду та зменшує кількість помилок. Програма також має вбудовану підтримку інструментів контролю версій (наприклад, Git), систем збірки (Maven, Gradle), тестування, профілювання та налагодження.

IntelliJ IDEA особливо ефективна при роботі з такими фреймворками, як Spring Boot [1], Hibernate, Java EE тощо. Вона доступна у двох версіях: Community Edition (безкоштовна) та Ultimate Edition (платна), яка надає повну підтримку для веброзробки, баз даних і корпоративних технологій.

2.2 Мови програмування

Мова програмування — це штучна мова, що використовується для створення програм та інструкцій, які комп'ютер може виконувати. Вона дозволяє програмістам писати алгоритми та передавати їх комп'ютеру для реалізації. Мови програмування мають власний синтаксис і граматику, які визначають правила написання програмного коду.

Існують різні типи мов: низькорівневі (наближені до машинного коду) і високорівневі (ближчі до людської мови, легші для сприйняття та написання). Серед найпоширеніших високорівневих мов — Java, Python, C++, JavaScript, C#, які використовуються для різних сфер: веброзробки, мобільних додатків, ігор, системного програмування, штучного інтелекту тощо.

Мови програмування можуть підтримувати різні парадигми: процедурну, об’єктно-орієнтовану, функціональну, логічну тощо. Кожна парадигма пропонує свій підхід до структурування програм. Вибір мови залежить від вимог проєкту, типу застосунку, команди розробників і цільової платформи.

2.**2.1 Java**

Java – це об’єктно орієнтована мова програмування загального призначення, яка була створена компанією Sun Microsystems у 1995 році (зараз розвивається компанією Oracle). Java є однією з найпопулярніших мов програмування у світі завдяки своїй стабільності, безпеці та кросплатформеності.

Java широко використовується в веброзробці, розробці Android-додатків, банківських системах, у створенні серверної частини додатків та Інтернету речей (IoT)[7].

Однією з головних переваг Java є її платформа-незалежність — програми, написані на Java, компілюються у байт-код, який виконується віртуальною машиною Java (JVM), що дозволяє запускати їх на будь-якому пристрої, де встановлена JVM. Це забезпечує високу гнучкість при розгортанні та обслуговуванні програмного забезпечення.

Java має потужну екосистему бібліотек і фреймворків, таких як Spring, Hibernate, Maven, які спрощують розробку складних програм. Мова підтримує багатопоточність, що дозволяє ефективно працювати з паралельними процесами. Завдяки сильній типізації, автоматичному управлінню пам’яттю (через збирач сміття) та широким можливостям налагодження, Java є надійним інструментом для створення критично важливих систем.

Активна спільнота розробників, регулярні оновлення та велика кількість навчальних ресурсів роблять Java відмінним вибором як для початківців, так і для досвідчених програмістів.

2.3 Веб-фреймворки

Веб-фреймворк - це програмний фреймворк, призначений для підтримки розробки веб-додатків, включаючи веб-сервіси, веб-ресурси та веб-API. Веб-фреймворки надають стандартний спосіб будувати та розгортати веб-додатки в Інтернеті.

Вони значно спрощують процес створення складних вебзастосунків, оскільки включають попередньо реалізовані компоненти, які обробляють типові завдання — маршрутизацію запитів, керування сесіями, роботу з базами даних, безпеку, автентифікацію, шаблонізацію HTML [18] тощо.

Існують фреймворки для різних мов програмування: наприклад, Spring Boot - для Java, Django - для Python, Laravel - для PHP, Express.js - для JavaScript (Node.js), Ruby on Rails - для Ruby.

Сучасні веб-фреймворки підтримують архітектурні підходи на кшталт MVC (Model-View-Controller), REST, а також можуть бути легко інтегровані з фронтенд-бібліотеками та фреймворками (React, Angular, Vue.js).

Завдяки фреймворкам веброзробники можуть зосередитися на бізнес-логіці застосунку, не витрачаючи час на реалізацію базової інфраструктури. Це сприяє швидшому запуску проєктів, покращенню якості коду та підвищенню безпеки.

2.3.1 Spring Boot

Spring Boot — це сучасний фреймворк для розробки вебзастосунків і мікросервісів на основі мови Java, що є частиною екосистеми Spring Framework. Його основна мета — спростити створення повноцінних Java-додатків за допомогою мінімальної конфігурації. Завдяки вбудованим серверам, таким як Tomcat або Jetty, Spring Boot дозволяє створювати самостійні (standalone) додатки, які можна запускати як звичайні Java-програми. Це значно полегшує процес розгортання, тестування та підтримки застосунків, адже не потребує встановлення окремого сервера або складного налаштування.

Фреймворк забезпечує автоматичну конфігурацію, що дозволяє розробникам зосередитися безпосередньо на реалізації бізнес-логіки, не витрачаючи час на ручне налаштування середовища. Завдяки тісній інтеграції з іншими компонентами екосистеми Spring, такими як Spring Data, Spring Security [4], Spring MVC, Spring Boot надає широкі можливості для реалізації складних

функціональностей у зручний і зрозумілий спосіб. Це робить його дуже привабливим для як початківців, так і досвідчених розробників.

Spring Boot активно використовується для створення REST API, роботи з базами даних, реалізації механізмів автентифікації та авторизації, ведення журналів подій, моніторингу стану системи та багатьох інших завдань. Його модульність і гнучкість дозволяють ефективно будувати масштабовані додатки, які легко розгортаються в локальному середовищі або хмарних інфраструктурах. Завдяки зручності налаштування, великій кількості документації та активній спільноті, Spring Boot став одним із найпопулярніших інструментів для побудови сучасного серверного програмного забезпечення.

Його філософія "конвенція понад конфігурацію" дозволяє значно скоротити час на початкову настройку проєкту. Багато рутинних завдань, які раніше потребували ручного втручання, тепер вирішуються автоматично. Крім того, Spring Boot спрощує тестування компонентів, завдяки чому забезпечується висока якість коду та стабільність додатка. Такий підхід дозволяє швидше запускати продукти в роботу, впроваджувати нові функції та підтримувати високий рівень надійності в експлуатації.

Завдяки поєднанню простоти, потужності та гнучкості, Spring Boot став стандартом де-факто для розробки сучасних вебзастосунків на Java. Він дозволяє зосередитись на створенні цінності для користувача, автоматизуючи багато технічних деталей та забезпечуючи стабільну основу для розвитку програмних рішень будь-якої складності.

2.3.2 Thymeleaf

Thymeleaf [6]— це сучасний Java-шаблонізатор для вебзастосунків, який дозволяє динамічно створювати HTML-сторінки на основі даних із сервера. Його основна роль — зв'язати бекенд (Java/Spring Boot) з фронтендом (HTML), забезпечуючи ефективне відображення динамічного контенту. Thymeleaf є рекомендованим шаблонізатором для Spring і чудово працює з формами, валідацією, даними з моделей.

Однією з головних переваг Thymeleaf є те, що HTML-код залишається валідним і читабельним навіть до обробки сервером. Це дозволяє дизайнерам і розробникам працювати з одним і тим самим кодом без потреби запускати сервер для перегляду шаблонів. Завдяки цьому шаблони можна легко переглядати в браузері на ранніх етапах розробки.

Thymeleaf повністю сумісний з HTML5 та дозволяє працювати з шаблонами, які виглядають як звичайні HTML-сторінки. Він додає власні атрибути (наприклад, th:text, th:each, th:if, th:object), які визначають, як повинна рендеритися сторінка з урахуванням динамічних даних з Java-коду[6].

Крім того, Thymeleaf підтримує фрагменти шаблонів (layout fragments), що дозволяє будувати багаторазово використовувані блоки інтерфейсу (наприклад, заголовки, меню, підвали), які можуть включатися у шаблони за допомогою th:insert або th:replace. Це значно покращує структуру проєкту, сприяє повторному використанню коду і спрощує його підтримку.

Завдяки легкій інтеграції з Spring Boot, Thymeleaf став популярним вибором для створення серверно-рендерених вебінтерфейсів, особливо для проєктів, де потрібна швидка розробка, прозора логіка та зручність підтримки.

2.4 Система управління базою даних

PostgreSQL, або просто Postgres, — це потужна система управління базами даних з відкритим вихідним кодом, яка поєднує в собі переваги реляційного підходу з гнучкістю об’єктно-орієнтованого програмування. Вона є однією з найнадійніших і найпопулярніших СУБД у світі, і активно використовується як у невеликих проєктах, так і у великих корпоративних рішеннях. Її архітектура дозволяє ефективно працювати з великими обсягами даних, забезпечуючи стабільність, масштабованість та високу продуктивність навіть за умов інтенсивного навантаження.

PostgreSQL повністю відповідає вимогам ACID, тобто гарантує атомарність, узгодженість, ізольованість і довговічність при виконанні транзакцій. Це означає, що всі операції з даними виконуються надійно і без втрат, навіть у випадку системних збоїв. Завдяки цьому PostgreSQL широко використовується у фінансових системах, державних інформаційних платформах, аналітичних сервісах та інших критично важливих сферах.

Однією з ключових переваг PostgreSQL є його потужний SQL-движок, що підтримує складні запити, збережені процедури, користувацькі функції, тригери, представлення та багато інших інструментів для тонкого управління даними. Крім класичних реляційних структур, PostgreSQL також дозволяє працювати з нереляційними типами даних — наприклад, JSON, XML або масивами, що особливо актуально для сучасних веб-застосунків і сервісів, які активно використовують формат JSON для обміну даними між клієнтом і сервером.

PostgreSQL має високу гнучкість у налаштуванні та розширенні. Завдяки підтримці модулів і розширень, таких як PostGIS для геопросторових даних або pg_stat_statements для аналітики запитів, систему можна адаптувати до найрізноманітніших потреб і галузей. Вона також дозволяє створювати власні функції за допомогою таких мов, як PL/pgSQL, Python, або навіть C, що відкриває широкі можливості для інтеграції з іншими компонентами програмного забезпечення.

Завдяки відкритому коду PostgreSQL має активну спільноту, яка постійно покращує систему, випускає оновлення безпеки та додає нові функції. Розробники й адміністратори цінують цю СУБД за її прозорість, потужність та можливість тонкої оптимізації під конкретні бізнес-потреби.

PostgreSQL залишається одним із найкращих рішень для тих, хто шукає надійну, масштабовану та функціонально багату систему керування базами даних у сучасному IT-середовищі.

2.5 Система контролю версіями

GitHub [3] - це онлайн-платформа для зберігання, управління та спільної роботи над програмним кодом, яка працює на основі системи контролю версій Git. Завдяки своїй функціональності, GitHub став однією з основних платформ для розробників у всьому світі, забезпечуючи ефективну взаємодію між учасниками проєктів, контроль змін у коді та централізоване зберігання даних.

Основним елементом GitHub є репозиторій — сховище, в якому зберігаються всі файли проєкту, історія змін, документація та інші ресурси. Користувачі можуть створювати як публічні, так і приватні репозиторії, що дозволяє працювати як над відкритими, так і над внутрішніми проєктами.

Однією з важливих функцій GitHub є можливість створення "форків" — копій чужих репозиторіїв, які можна редагувати та розвивати незалежно. Після внесення змін користувач може надіслати пропозицію щодо їх включення в основний проєкт через механізм pull request. Це забезпечує гнучку та прозору співпрацю між розробниками та сприяє активному розвитку відкритого програмного забезпечення. GitHub також надає інструменти для організації командної роботи, зокрема систему задач (issues), обговорень, вікі-документації та інтеграції з системами CI/CD, що дозволяє автоматизувати процеси тестування, збірки й розгортання програм.

Завдяки цьому платформа широко використовується не лише для особистих проєктів, а й у професійному середовищі великих компаній, де важливо забезпечити контроль якості коду та ефективну співпрацю між командами. Крім того, GitHub активно підтримує спільноти розробників, сприяючи обміну досвідом, публікації відкритого коду та залученню нових учасників до різноманітних проєктів. [3].

2.6 Інструменти проєктування інтерфейсу користувача

Figma [8] - це онлайн-інструмент для створення дизайну інтерфейсів, який надає можливість командам разом працювати над макетами та прототипами в реальному часі. Запущений у 2012 році Діланом Філдом і Еваном Воллесом, цей сервіс швидко здобув популярність серед дизайнерів завдяки своїй простоті, функціональності та доступності з будь-якого пристрою через браузер.

Головна перевага Figma полягає в тому, що вона працює повністю у веб-середовищі, не вимагаючи встановлення програм на комп’ютер. Це дозволяє членам команди з різних куточків світу легко долучатися до проєкту, редагувати макети та обговорювати зміни прямо в інтерфейсі інструменту.

Figma також пропонує вбудовані можливості для створення інтерактивних прототипів з анімацією та мікровзаємодіями, що дає змогу протестувати користувацький досвід без потреби у додатковому ПЗ.

Окрім цього, інструмент підтримує розробку дизайн-систем: користувачі можуть зберігати стилі, компоненти, кольорові палітри та інші елементи в централізованих бібліотеках, що забезпечує цілісність і повторне використання дизайну у різних проєктах.

3 ПРОЕКТУВАННЯ

3.1 Проектування архітектури застосунку

Застосунок реалізовано за клієнт-серверною архітектурною моделлю, яка забезпечує чітке розмежування обов'язків між клієнтською та серверною частинами системи.[19] Такий підхід сприяє кращій масштабованості, підтримуваності та дозволяє окремо розвивати функціональність інтерфейсу та бізнес-логіку. Клієнтська частина створена з використанням HTML, CSS[10], JavaScript і шаблонізатора Thymeleaf, який дає змогу динамічно формувати сторінки на основі даних із сервера. Це значно спрощує обробку інформації на фронтенді, оскільки більшість логіки генерування інтерфейсу переноситься на сервер. Завдяки цьому користувач взаємодіє з інтуїтивно зрозумілим інтерфейсом, не потребуючи глибоких технічних знань.

Серверна частина розроблена з використанням фреймворку Spring Boot, що входить до екосистеми Spring і дозволяє швидко створювати веб-застосунки з чіткою структурою. Сервер відповідає за обробку запитів від клієнта, виконання логіки перевірки відповідей, нарахування балів, а також збереження результатів у базі даних. Уся інформація, включно з тестами, запитаннями, відповідями, результатами й користувачами, зберігається у реляційній базі даних PostgreSQL[2], що гарантує надійність, транзакційність і цілісність даних. Завдяки використанню Spring Data JPA спрощується взаємодія з базою даних, а доступ до даних здійснюється через зручні репозиторії.

Застосунок підтримує розмежування прав доступу до функціоналу. Користувачі мають визначені ролі, зокрема ролі адміністратора й звичайного користувача. Авторизація та аутентифікація реалізовані через Spring Security, що дозволяє забезпечити захищений доступ до важливої інформації, запобігти несанкціонованим змінам даних і обмежити функціонал відповідно до ролі

користувача. Усі паролі зберігаються в зашифрованому вигляді, а система передбачає механізми сесій та перевірку дозволів для виконання певних дій.

Процес проходження тесту реалізовано таким чином, що користувач бачить одне запитання на сторінці, має можливість обрати відповідь і перейти до наступного. Система може враховувати як запитання з одним правильним варіантом, так і з кількома, а також підтримує відкриті відповіді. Після завершення тесту результати автоматично обробляються, і користувач отримує загальну оцінку, а також зворотний зв’язок із зазначенням правильних і неправильних відповідей. Для адміністратора доступна статистика по кожному тесту, включаючи середні бали, кількість спроб і поширені помилки.

Інтерфейс побудований таким чином, щоб мінімізувати кількість дій, необхідних для керування системою. Адміністратор може створювати нові тести, додавати або змінювати запитання, редагувати варіанти відповідей і бачити результати всіх користувачів. Окремим блоком реалізовано функціонал експорту результатів у формат Excel, що дозволяє швидко аналізувати дані, формувати звіти та друкувати підсумкові відомості.

Під час реалізації враховано вимоги до стабільності та безпеки. Система передбачає захист від повторного проходження тестів, контроль часу, обмеження доступу до правильних відповідей до моменту завершення тестування. Для уникнення навантаження під час масових сесій реалізовано асинхронну обробку деяких запитів, а також розроблено можливість масштабування серверної частини у разі підключення більшої кількості користувачів.

Завдяки обраним технологіям і архітектурним підходам застосунок є гнучким, стабільним і придатним для тривалого використання в освітньому середовищі, зокрема для підвищення кваліфікації державних службовців. У перспективі його можна доповнити адаптивними тестами, мобільною версією або модулем аналітики, не змінюючи базову структуру застосунку.

3.2 Архітектура клієнтської частини

Проєктування клієнтської частини веб-застосунку було зосереджено на забезпеченні зручності взаємодії користувача з системою, логічній структурі інтерфейсу та відповідності функціональним ролям. Основна увага приділялась побудові чіткої навігації, модульності інтерфейсних елементів і забезпеченню динамічного відображення інформації відповідно до дій користувача.

Під час проектування було визначено ключові шаблони сторінок, які відображають основні етапи роботи користувача із застосунком — реєстрація, автентифікація, проходження тестування, перегляд результатів, адміністративне керування. Кожна з цих сторінок реалізовує окремий функціональний блок, що спрощує логіку переходів між ними та сприяє підвищенню зручності користування.

Інтерфейс було розділено на універсальні компоненти, такі як заголовок, меню навігації, блок повідомлень, які проектувалися як багаторазові модулі для повторного використання на різних сторінках. Це дозволило забезпечити єдність дизайну, спростити оновлення інтерфейсу та забезпечити зручність підтримки.

Особливу увагу при проектуванні було приділено гнучкості інтерфейсу - реалізовано умови відображення елементів у залежності від ролі користувача[20] (звичайний користувач або адміністратор), а також надано можливість динамічного оновлення контенту сторінки відповідно до поточного стану (наприклад, питання тесту або результати проходження).

Клієнтська частина тісно інтегрована з серверною логікою: шаблони інтерфейсу були спроектовані таким чином, щоб легко приймати та відображати динамічні дані, які надходять із контролерів серверної частини. Це дозволило реалізувати чіткий розподіл обов’язків - клієнтська частина відповідає за візуалізацію, а серверна за обробку даних, що позитивно впливає на масштабованість та підтримуваність системи.

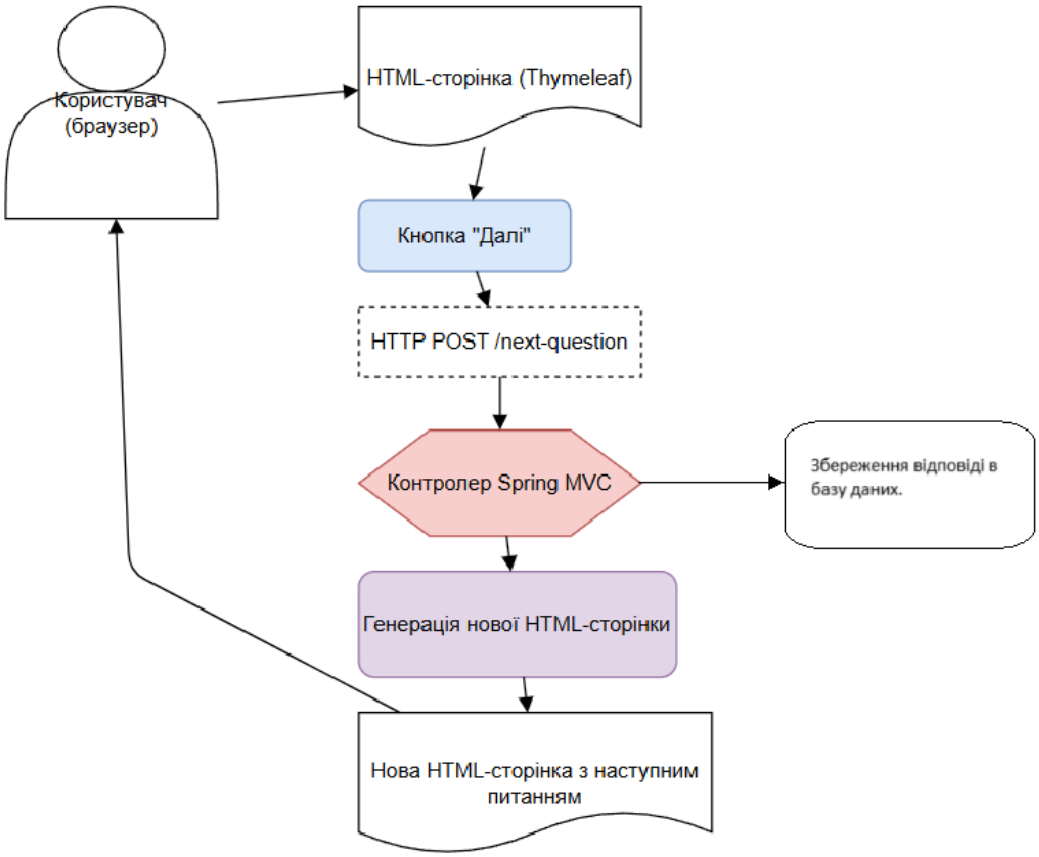


Рис 3.1 Схема клієнтської частини застосунку

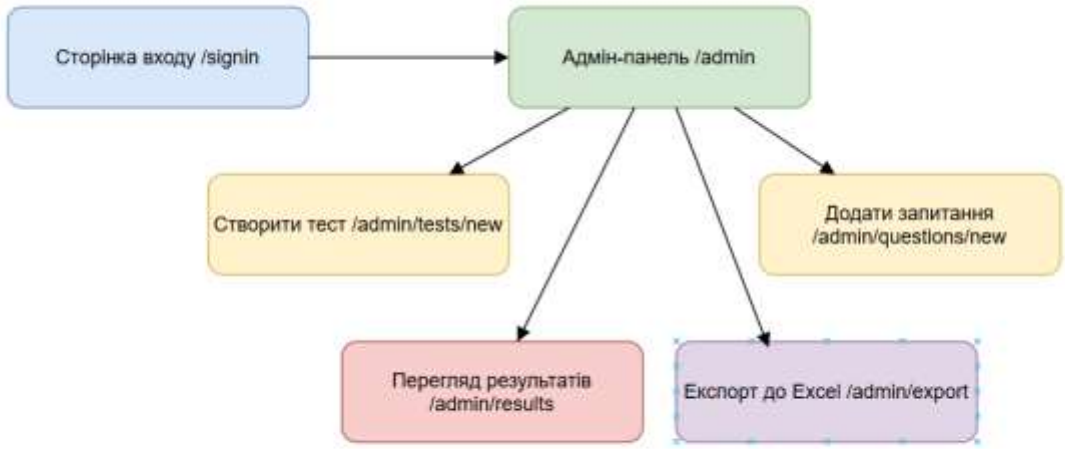


Рис 3.2 Схема клієнтської частини андміністратора

3.3 Архітектура серверної частини

Серверна частина проєкту побудована на основі фреймворку Spring Boot, що забезпечує зручну модульну структуру та високий рівень масштабованості. Spring Boot спрощує налаштування, запуск та конфігурацію додатків завдяки концепції "конвенції замість конфігурації", що дозволяє зосередитися на розробці логіки, а не на інфраструктурних питаннях.

Архітектура реалізована за принципом шаруватої моделі (layered architecture), яка чітко розділяє відповідальності між різними компонентами застосунку:

- Controller (Контролери): відповідають за обробку HTTP-запитів, що надходять від клієнта. Контролери приймають запити, викликають відповідні сервіси, формують модель з даними і передають її у шаблон Thymeleaf. Це забезпечує логіку переходу між сторінками, обробку форм, авторизацію користувачів і відображення результатів.

- Service (Сервіси): містять бізнес-логіку застосунку. Саме в сервісному шарі відбувається обробка даних, перевірка правильності відповідей, підрахунок балів, формування результатів тестування, логіка доступу до тестів і ролей користувачів. Сервіси ізольовані від контролерів, що спрощує їхнє тестування, повторне використання та підтримку.

- Repository (Репозиторії): реалізують доступ до бази даних за допомогою Spring Data JPA. Репозиторії взаємодіють з об'єктами сутностей (Entity), виконують збереження, пошук, оновлення та видалення даних. Робота з базою PostgreSQL відбувається через об'єктно-реляційне відображення (ORM), що дозволяє розробникам працювати з об'єктами Java, не турбуючись про SQL-запити.

- Model (Моделі / Сутності): представляють структуру таблиць бази даних. Кожна сутність описується як Java-клас з анотацією @Entity і містить поля, які

відповідають колонкам у таблицях PostgreSQL. Наприклад, модель Question описує текст питання, варіанти відповідей і правильну відповідь, а модель Result -інформацію про користувача, питання, його відповідь і набрані бали.

Завдяки такій структурі серверна частина легко розширюється, підтримується та тестується. Spring Boot також дозволяє інтегрувати інші компоненти, такі як безпека (Spring Security), логування, валідацію, кешування та REST API для взаємодії з клієнтськими застосунками.

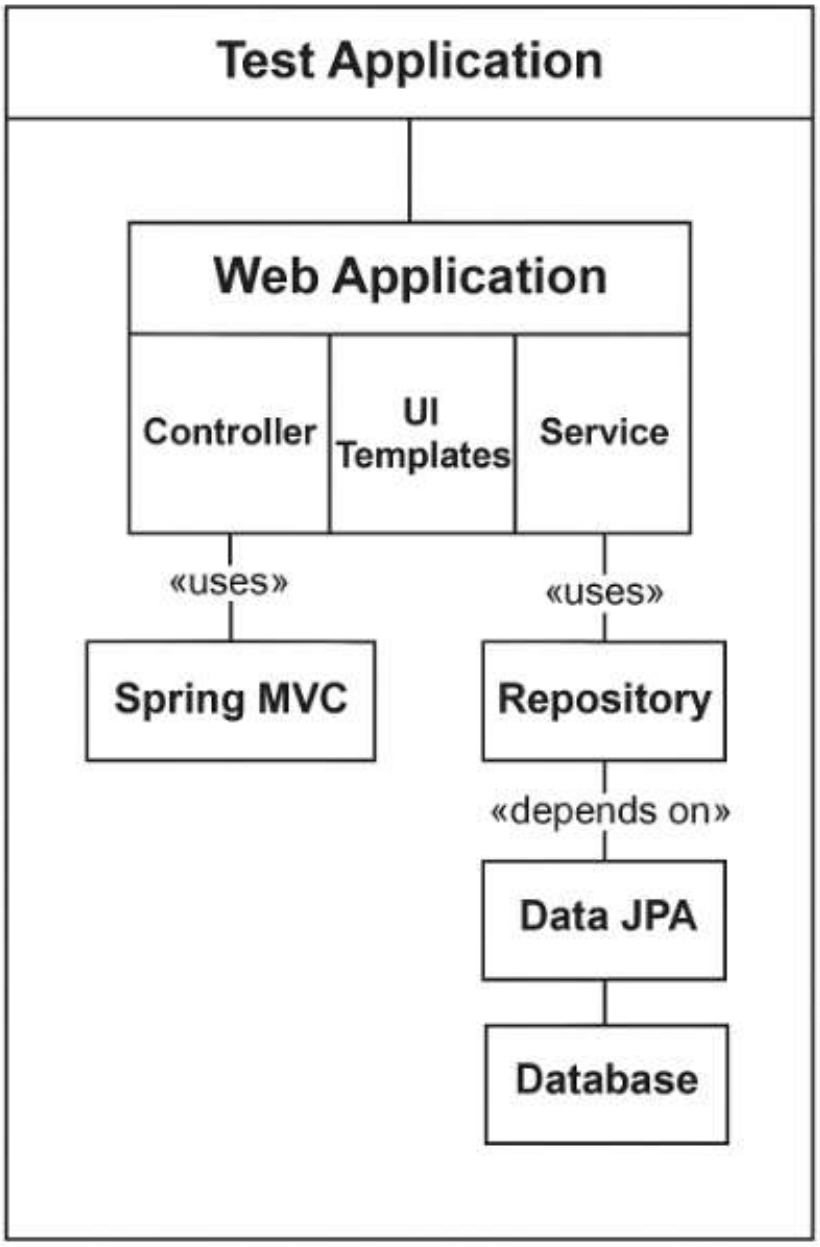


Рис 3.3 Діаграма компонентів

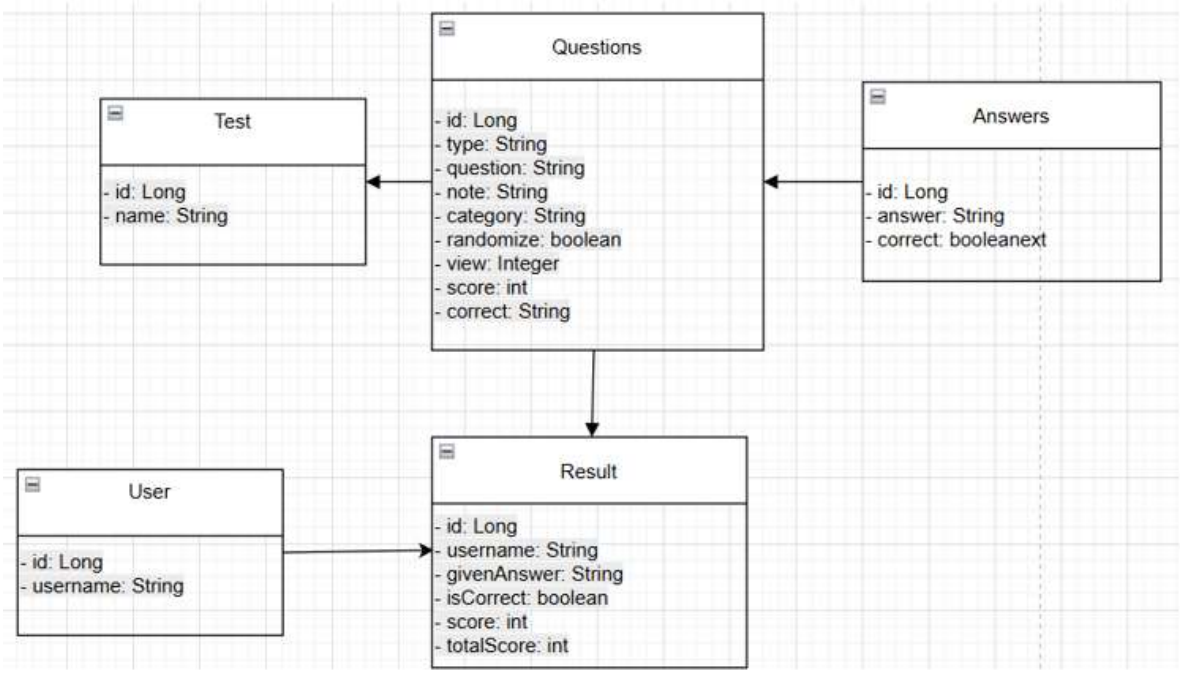


Рис 3.4 Діаграма класів

3.4 Архітектура бази даних

Архітектура бази даних проєкту побудована на основі реляційної моделі, яка реалізується за допомогою системи управління базами даних PostgreSQL. Цей підхід дозволяє ефективно організувати структуру зберігання даних, забезпечити їхню цілісність, швидкий доступ, масштабованість та зручне керування. PostgreSQL обрано через її стабільність, гнучкість, підтримку складних запитів, транзакцій, індексів та широкі можливості розширення.

База даних складається з декількох основних таблиць, які відповідають ключовим сутностям проєкту, таким як тести, питання, варіанти відповідей, користувачі, результати тестування тощо. Структура таблиць ретельно продумана відповідно до предметної області онлайн-тестування та реалізована у вигляді Java-класів з анотаціями JPA (@Entity, @Column, @Id, @ManyToOne, @OneToMany, @ManyToMany тощо) за допомогою Spring Data JPA. Це

забезпечує зручне об’єктно-реляційне відображення (ORM), при якому розробники працюють з об'єктами Java, а не з SQL-запитами безпосередньо.

Кожна таблиця має свій набір колонок, що визначають типи даних, правила зберігання та зв’язки з іншими таблицями. Відносини між таблицями побудовані згідно з логікою предметної області:

- "один до багатьох" (наприклад, один тест містить багато питань),

- "багато до одного" (кожне питання належить одному тесту),

- "багато до багатьох" (наприклад, користувачі можуть проходити багато тестів, і один тест можуть проходити багато користувачів — реалізується через проміжну таблицю).

Основні таблиці бази даних:

- users - зберігає облікові дані зареєстрованих користувачів, їхні ролі та особисту інформацію.

- tests - містить загальну інформацію про тести: назву, опис, час на виконання.

- questions - включає текст питань, тип питання (один або кілька правильних варіантів), зв'язок із конкретним тестом.

- answers - варіанти відповідей до кожного питання, включаючи ознаку правильності.

- results - результати проходження тестів із фіксацією користувача, тесту, набраних балів, часу проходження тощо.

Загалом, реляційна база даних на PostgreSQL дозволяє ефективно обробляти та зберігати великі обсяги інформації, забезпечуючи надійність, швидкодію та масштабованість веб-додатку. Такий підхід дає змогу легко розширювати систему в майбутньому, додаючи нові функціональні модулі без значної перебудови базової структури.

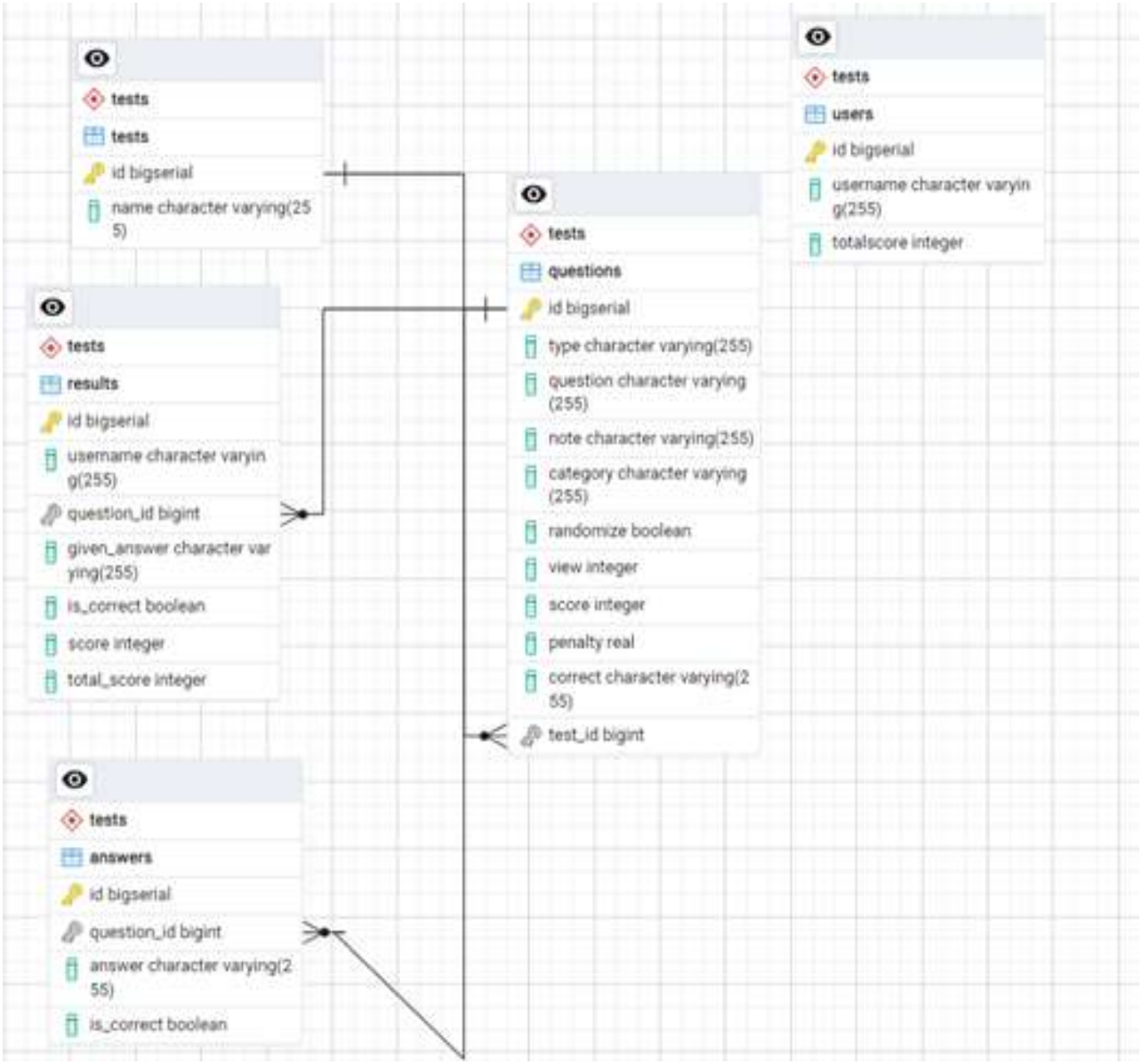


Рис 3.3 Загальна схема бази даних

4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

4.1 Дизайн інтерфейсу

Figma - це сучасний і зручний інструмент для створення дизайну інтерфейсів користувача та розробки прототипів, який дає змогу працювати над макетами інтерактивно та спільно в реальному часі. Нижче подано основні етапи, які були використані під час прототипування UI за допомогою Figma:

Створення нового проєкту

Спершу необхідно зареєструватися або увійти у свій обліковий запис на платформі Figma.

– Для початку роботи натискаємо "New File" на головному екрані, що відкриває новий дизайн-файл.

Додавання фреймів

Фрейми у Figma виконують роль контейнерів, у яких розміщуються елементи інтерфейсу. Вони можуть представляти окремі сторінки або частини макету.

- Щоб створити фрейм, потрібно обрати інструмент "Frame" на панелі інструментів зліва або натиснути клавішу F, після чого позначити область на робочому полі.

- Розміри фрейму можна обрати зі стандартних варіантів (для мобільних, планшетів, десктопів) або задати вручну у правій панелі властивостей.

- За допомогою інструментів, таких як "Rectangle", "Line", "Ellipse" тощо, створюються базові візуальні елементи інтерфейсу.

- Для вставки тексту використовується інструмент "Text" (або клавіша T).

- Зображення та іконки можна імпортувати з комп’ютера або використовувати ресурси з бібліотеки Figma. Компоненти дають змогу створювати повторювані елементи інтерфейсу, які легко оновлюються в усіх місцях використання.

- Щоб створити компонент, потрібно виділити бажані елементи, натиснути праву кнопку миші та обрати "Create Component", або скористатися комбінацією клавіш Ctrl+Alt+K.

- Створені компоненти можна повторно використовувати, перетягуючи їх із панелі активів (Assets) на робоче поле.

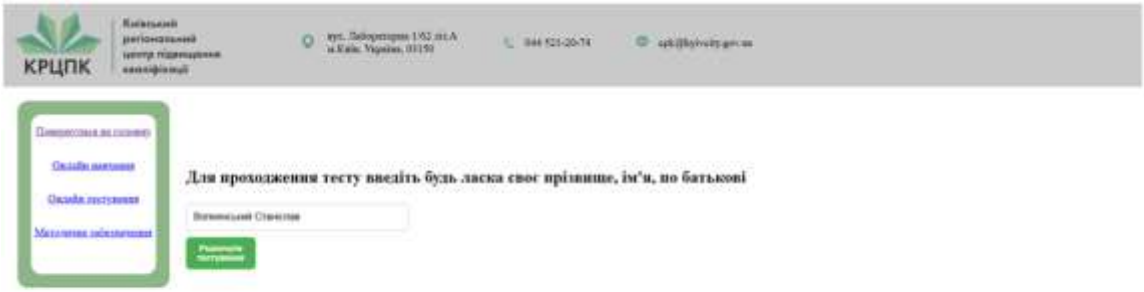


Рис 4.1 Екранна форма протопиту застосунка

4.2 Реалізація клієнтської частини

Для реалізації клієнтської частини проекту використовується JavaScript у поєднанні з HTML та CSS. У якості шаблонізатора для динамічного формування веб-сторінок застосовується Thymeleaf. Використання цього шаблонізатора дозволяє зручно інтегрувати динамічні дані з серверної частини на веб-інтерфейс користувача.

Після ініціалізації Spring Boot-проекту, структура каталогів проекту має такий вигляд:

- Каталог /src/main/resources/templates/ – тут розміщуються HTML-шаблони, що використовують можливості Thymeleaf. Кожен шаблон відповідає окремому веб-інтерфейсу

- static/ – призначений для розміщення статичних ресурсів

- js/ – тут знаходяться JavaScript-файли, що відповідають за динамічну поведінку елементів на веб-сторінках

- css/ – стилі для оформлення веб-інтерфейсу.

Основна частина для створення динамічної поведінки елементів написана за допомогою Thymeleaf.

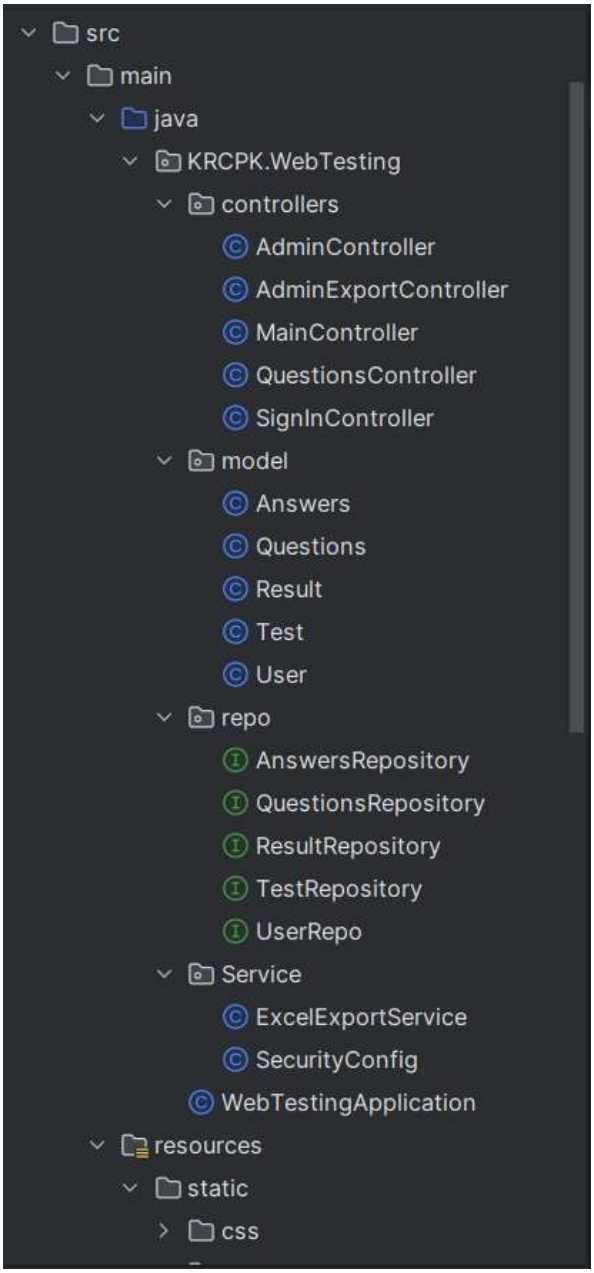


Рис 4.2 Приклад структури проєкту

Другим кроком у реалізації клієнтської частини проєкту є створення ключових сторінок (шаблонів) та підключення до них необхідних скриптів і стилів.

В проєкті використовується Thymeleaf – серверний Java-шаблонізатор, який дозволяє інтегрувати динамічні дані з серверної частини (Spring Boot) безпосередньо у HTML-розмітку.

Створення HTML-шаблонів

HTML-шаблони розміщуються у каталозі: /src/main/resources/templates/

Кожен шаблон відповідає окремому функціональному компоненту додатку (наприклад: сторінка входу, сторінка тестування, сторінка перегляду результатів, сторінка адміністратора).

- Файл HTML містить розмітку веб-сторінки та спеціальні Thymeleaf-атрибути (наприклад, th:each, th:value, th:if) для динамічного відображення даних.

- Файл стилів CSS визначає зовнішній вигляд компоненту: кольори, шрифти, розташування елементів.

- Файл images зберігає в собі всі картинки які використовуються в веб сторінках.

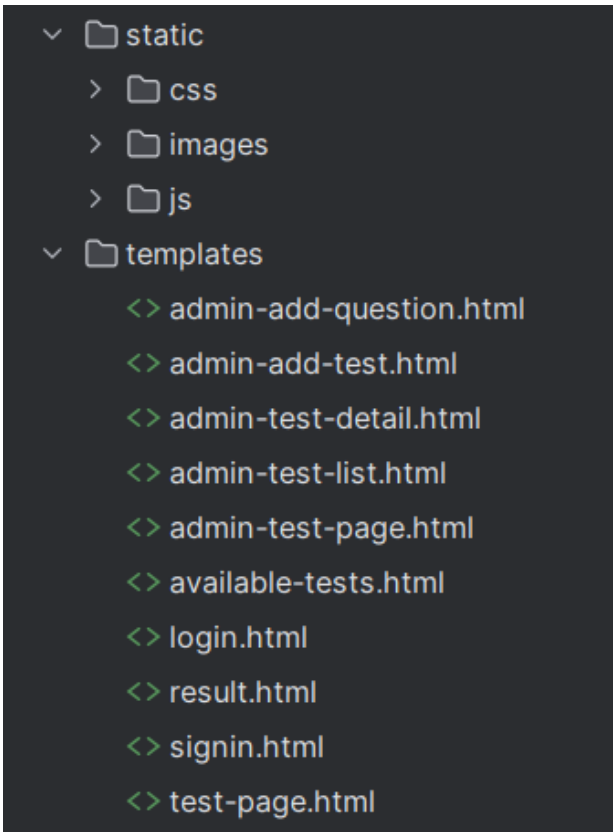


Рис 4.3 Приклад створених HTML – шаблонів

4.3 Реалізація серверної частини

Серверна частина застосунку розроблена з використанням фреймворку Spring Boot, який забезпечує гнучку платформу для створення сучасних веб-застосунків та RESTful API. Завдяки широким можливостям автоматичної конфігурації, високій продуктивності та безпеці, Spring Boot є ідеальним інструментом для розробки серверної логіки.

На початковому етапі реалізації були створені ключові компоненти серверної частини: контролери, сервіси, репозиторії та модельні класи. Структура проєкту була чітко організована: контролери розміщено у відповідному пакеті, що відповідає за обробку запитів, модельні класи визначають структуру даних, а репозиторії забезпечують взаємодію з базою даних.

Контролери є центральною ланкою, яка з’єднує користувача із застосунком. Вони керують логікою обробки запитів, маршрутизують їх до відповідних сервісів, обробляють дані та передають відповіді назад до користувача. Кожен контролер відповідає за окремий функціональний блок: управління тестуванням, автентифікація користувачів, обробка результатів тощо.

На наступному етапі реалізовано функціонал взаємодії користувача з системою тестування. Користувач має змогу проходити тести, а система — відстежувати, які запитання йому вже показано, які відповіді обрано, та відповідно реагувати на його дії. У разі введення даних під час автентифікації вони перевіряються, після чого користувач отримує доступ до відповідного тесту.

Особлива увага приділялася логіці обробки результатів тестування. Після завершення тесту система підраховує бали, визначає правильність відповідей та зберігає результати до бази даних. Це забезпечує можливість подальшого аналізу результатів кожного користувача.

Адміністративний функціонал є важливою складовою серверної частини. Адміністратор має змогу входити в систему через окремий інтерфейс, отримувати доступ до внутрішніх сторінок, керувати тестами, створювати нові питання та переглядати статистику. Доступ до адміністративного функціоналу контролюється системою безпеки, що гарантує його захищеність.

Для ефективного управління контентом адміністратор може створювати нові тести, редагувати вже існуючі, а також додавати до них запитання. Створення пов’язаної структури між тестами та запитаннями забезпечує гнучку і логічну організацію контенту.

Рис 4.4 Блок-схема обробки результатів тестування

Завершальним етапом реалізації серверної частини стало впровадження функції експорту результатів тестування у форматі Excel. Цей функціонал дозволяє отримати структуровані дані про відповіді користувачів, їхню успішність та загальну статистику. Така інформація є цінною для звітності, аналізу успішності проходження тестів, а також для подальшого прийняття рішень на основі отриманих даних.

Загалом, серверна частина проєкту реалізована з урахуванням сучасних вимог до безпеки, масштабованості та зручності використання, що дозволяє забезпечити стабільну та ефективну роботу системи в умовах реального навантаження.

4.4 Завантаження проєкту на GitHub

Протягом всього процесу розробки виконувалося завантаження всіх файлів та вихідного коду проєкту на GitHub — популярний сервіс, який використовувався для контролю версій та спільної розробки програмного забезпечення. Завдяки GitHub забезпечувалася можливість відстеження змін, управління різними версіями проєкту, а також зручний доступ до коду для інших розробників і користувачів. Це суттєво полегшило процес підтримки та подальшого розвитку проєкту, дозволило зберігати історію змін і швидко повертатися до попередніх версій за потреби.

Для завантаження проєкту було створено репозиторій на GitHub з відповідним описом. Далі у веб-інтерфейсі GitHub було обрано опцію Add file та пункт Upload files. Після вибору всіх потрібних файлів вони були додані до репозиторію. Для завершення процесу було додано опис коміту з поясненням внесених змін та натиснуто кнопку Commit changes, після чого всі файли проєкту стали доступними для подальшої роботи та контролю версій.

4.5 Тестування застосунку

Одним із ключових етапів у процесі розробки веб-застосунку для тестування державних службовців є тестування, що дозволяє перевірити коректність реалізації функцій, стабільність роботи системи, а також відповідність функціональним та нефункціональним вимогам. У цьому проєкті особливу увагу було приділено функціональному, інтеграційному, ролевому та регресійному тестуванню. Оскільки застосунок має розгалужену логіку керування доступом і різні ролі користувачів, критично важливим було перевірити, чи всі компоненти працюють у взаємодії без збоїв, а права доступу обмежені відповідно до ролі.

Функціональне тестування було спрямоване на перевірку основних можливостей системи: реєстрації та автентифікації користувачів, створення та редагування тестів, проходження тестування, а також відображення та оцінювання результатів. Було проведено серію перевірок, що підтвердили правильність обробки форм введення, валідації даних, розрахунку правильних відповідей та відображення результатів. Наприклад, при створенні тесту система коректно зберігала всі питання, варіанти відповідей та позначала правильні з них, а при проходженні — вірно підраховувала бали й формувала зведену статистику.

Інтеграційне тестування охоплювало взаємодію між модулями автентифікації, керування тестами та результатами. Було перевірено передачу даних між репозиторіями, сервісами та контролерами, а також узгодженість відображення даних на сторінках за допомогою шаблонізатора Thymeleaf. Наприклад, після створення тесту адміністратором він відображався у списку доступних тестів у звичайного користувача, а після проходження - з’являвся у звіті з оцінкою. Тестування показало, що дані зберігаються без втрат, логіка відображення працює правильно, а некоректні дії (наприклад, доступ до результатів без проходження тесту) блокуються.

Окрему увагу приділено тестуванню ролей і доступу. Система має чітке розділення прав між адміністраторами та звичайними користувачами. Адміністратор має змогу створювати, редагувати й видаляти тести, переглядати всі результати та керувати користувачами. Звичайний користувач — лише проходити тести та переглядати власні результати. Також тестувалася коректна обробка переходів між сторінками залежно від ролі користувача.

Регресійне тестування проводилося після кожного етапу внесення змін у код — особливо під час оновлень логіки збереження даних, авторизації або відображення результатів. Повторно перевірялися всі ключові функції, щоб упевнитися, що нові зміни не порушили вже реалізовані можливості. Було автоматизовано перевірку завантаження сторінок, наявності елементів інтерфейсу та обробки запитів на сервер. Завдяки цьому вдалося вчасно виявити та усунути кілька помилок, пов’язаних із неправильним перенаправленням користувачів після логіну та некоректною обробкою незбережених форм.

За підсумками тестування встановлено, що:

- Усі функції застосунку відповідають поставленим вимогам.
- Розмежування доступу реалізоване коректно.
- Інтерфейс стабільно працює при багатьох одночасних запитах.
- Дані зберігаються без втрат та помилок.
- Система є надійною, зручною у користуванні та готовою до реального застосування.

Таким чином, результати тестування підтвердили працездатність системи, її стабільність та відповідність вимогам до систем онлайн-тестування державних службовців.

ВИСНОВКИ

У процесі виконання проєкту було досягнуто поставлених цілей та виконано всі визначені задачі. Під час аналізу існуючих аналогів систем тестування було досліджено різні підходи до реалізації подібних застосунків, що дозволило виявити їх переваги та недоліки. Це, у свою чергу, допомогло визначити ключові вимоги до власного проєкту, орієнтованого на зручність користування, надійність та ефективність.

Була розроблена та реалізована структура бази даних, яка забезпечує зберігання тестів, питань, варіантів відповідей та результатів користувачів, що дозволяє багаторазово проходити тести та зберігати прогрес кожного користувача. Реалізований механізм підрахунку балів та збереження відповідей користувачів забезпечує коректність оцінки тестування та формування статистики результатів.

Також було створено функціонал експорту результатів тестування у форматі Excel, що спрощує аналіз даних, їхнє подальше зберігання та друк. Вхід адміністратора дозволяє редагувати тести, додавати нові та контролювати зміст тестових сесій, що розширює можливості системи для управління навчальним процесом. Окрім цього, користувачі отримують можливість переглядати результати своїх тестів, включаючи неправильні відповіді та загальний бал, що сприяє більш ефективному навчанню та аналізу власних помилок.

Таким чином, проєкт дозволяє автоматизувати процес тестування та спрощує роботу як користувачів, так і адміністраторів, забезпечуючи стабільність, надійність та масштабованість розробленого застосунку.

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей:

1. Волнянський С.Р., Яскевич В.О Визначення вимог до сайту тестування публічних службовців. Матеріали: VI Всеукраїнська науково-

технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». 25.04.2025, ДУІКТ, Київ, Україна, С. 105-108.

2. Волнянський С.Р., Яскевич В.О. Переваги використання Spring Boot у розробці вебзастосунку для онлайн-тестування публічних службовців. Матеріали: VI Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». 25.04.2025, ДУІКТ, Київ, Україна, С. 109-111.

ПЕРЕЛІК ПОСИЛАНЬ

1. Spring. Spring Boot Official Documentation. (дата звернення 16.03.2025) URL https://spring.io/projects/spring-boot

2. PostgreSQL Global Development Group. PostgreSQL Official Website.(дата звернення 15.03.2025)
URL: https://www.postgresql.org

3. GitHub Docs. Managing your repository's settings.(дата звернення 21.03.2025)
URL: https://docs.github.com/en/repositories/configuring-branches-and-merges-in-your-repository/configuring-protected-branches

4. Spring Security Reference. (дата звернення 06.05.2025)
URL: https://docs.spring.io/spring-security/reference/

5. JavaScript: JavaScript Guide / Mozilla Developer Network. (дата звернення 15.03.2025)
URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide

6. Thymeleaf. Thymeleaf Documentation. (дата звернення 15.03.2025)
URL: https://www.thymeleaf.org

7. Oracle. Java Documentation.(дата звернення 16.04.2025)
URL: https://docs.oracle.com/javase

8. Figma. Офіційна документація (дата звернення 10.03.2025)
URL: https://help.figma.com/hc/en-us

9 Patel D., Applications, and Challenges. International Journal of Computing and Digital Systems. 2023. Page. 851-868.(дата звернення 25.04.2025)
URL: https://doi.org/10.12785/ijcds/130168

10. CSS: Cascading Style Sheets / Mozilla Developer Network. (дата звернення 15.03.2025)
URL: https://developer.mozilla.org/en-US/docs/Web/CSS

11. Walls C. Spring in Action. 6th ed. Manning Publications, 2022. Page 520

12. Cosmina I. Spring 5 Design Patterns Publishing, 2021. Page 392

13. Литвинова С.Г. Хмарні сервіси Office 365 для вчителів: навчальний посібник. Київ: Компринт, 2021. 256 с.

14. Owasp Top 10 - 2021: The Ten Most Critical Web Application Security Risks. Open Web Application Security Project. (дата звернення 28.04.2025) URL: https://owasp.org/Top10/

15. IntelliJ IDEA: The Java IDE for Professional Developers. JetBrains.(дата звернення 15.03.2025)
URL: https://www.jetbrains.com/idea/

16. Morville P., Rosenfeld L., Arango J. Information Architecture: For the Web and Beyond. 5th ed. O'Reilly Media, 2023. 528 p. (дата звернення 02.03.2025)

17. Moving Forward to New Educational Realities in the Digital Era–Report of EDUsummIT 20222-2023 (дата звернення 03.05.2025)

18. HTML: Hypertext Markup Language / Mozilla Developer Network.(дата звернення 14.03.2025)
URL: https://developer.mozilla.org/en-US/docs/Web/HTML

19. Global EdTech Market Report 2023-2030: Focus on Digital Learning Platforms, Assessment Tools, and Learning Management Systems. HolonIQ.(дата звернення 24.05.2025)
URL: https://www.holoniq.com/platform

20. Stallings W., Brown L. Computer Security: Principles and Practice. 5th ed. Pearson, 2023. Page 800

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка програмного забезпечення для підтримки процесів тестування в Київському регіональному центрі підвищення кваліфікації

Виконав студент 4 курсу
Групи ПД-43
Волнянський Станіслав Русланович
Керівник роботи
к.т.н., доц., доцент кафедри ПІЗ Яскевич Владислав Олександрович
Київ – 2025

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Метою роботи:** допомога в організації та проведенні онлайн-тестування державних службовців з метою перевірки рівня їхніх знань та навичок.
- **Об'єкт дослідження:** процес підвищення кваліфікації державних службовців із застосуванням інформаційних технологій.
- **Предмет дослідження:** застосунок пов'язаний з тестуванням та оцінюванням знань державних службовців у системі підвищення кваліфікації.

2

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. **Ознайомитися з існуючими аналогами систем тестування**, оцінити їх переваги та недоліки з точки зору функціональності, зручності користування та надійності.
2. **Розробити структуру бази даних для збереження тестів, питань, варіантів відповідей та результатів користувачів** із можливістю багаторазового проходження тестів.
3. **Реалізувати механізм збереження відповідей користувача та підрахунку результатів**, з урахуванням правильності відповідей та нарахуванням балів.
4. **Розробити функціонал експорту результатів тестування у форматі Excel** для подальшого аналізу, зберігання або друку.
5. **Розробити функціонал входу адміністратора**, для редагування та створення нових тестів.
6. **Розробити можливість перегляду результатів тестування користувача** в вигляді неправильних відповідей та загального балу.

3

АНАЛІЗ АНАЛОГІВ

Додаток	Google Forms	Moodle	Testepk
Зручність створення тестів	+	-	+
База даних	-	+	+
Можливість багаторазового проходження	-	+	+
Автоматичний підрахунок балів	+	+	+
Експорт результатів(Excel)	-	+	+
Аналіз неправильних відповідей для користувача	-	+	+
Інтуїтивний інтерфейс	+	-	+

4

ВИМОГИ ДО ДОДАТКУ

Функціональні вимоги:

1. Реєстрація та автентифікація користувачів;
2. Управління даними тестів користувачів;
3. Створення та редагування тестів;
4. Проходження тестування;
5. Оцінювання результатів;
6. Можливість входу з правами адміністратора.

Нефункціональні вимоги:

1. Безперебійне функціонування, збереження даних;
2. Захист персональних даних і доступу до системи;
3. Підтримка різних браузерів;
4. Швидке завантаження сторінок за допомогою «жадібного завантаження».

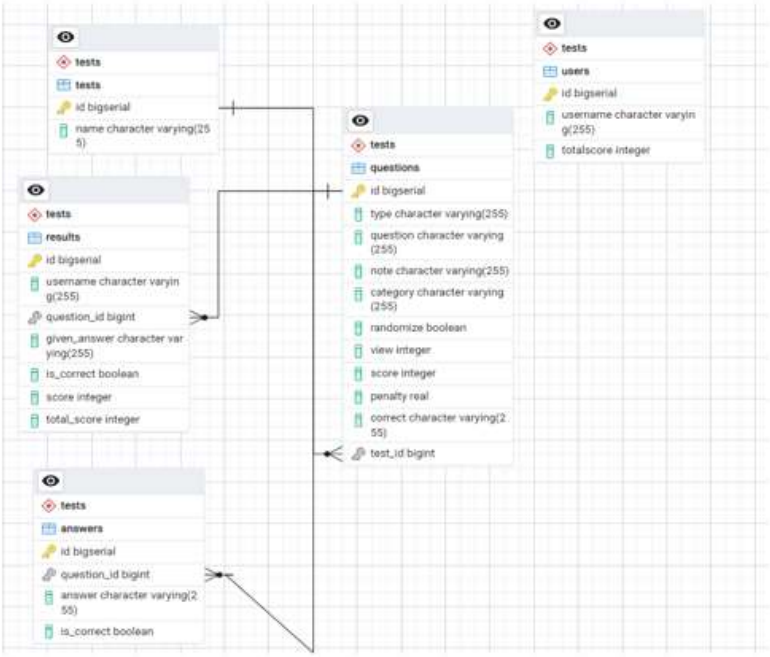
5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

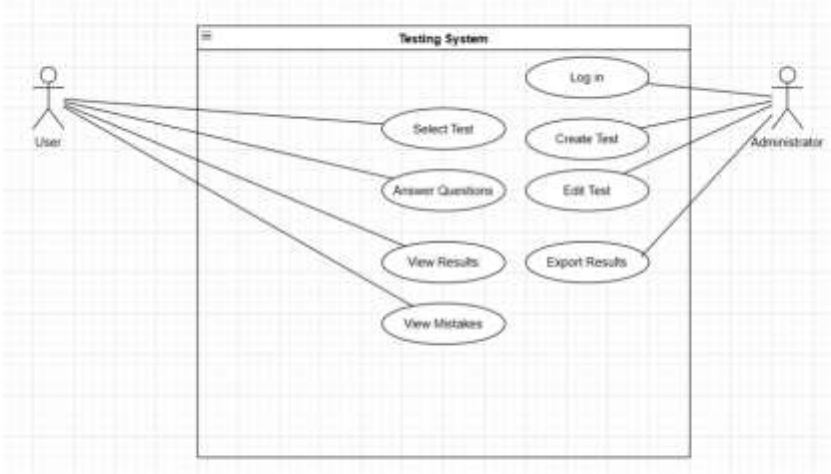
Intellij.IDEA

6

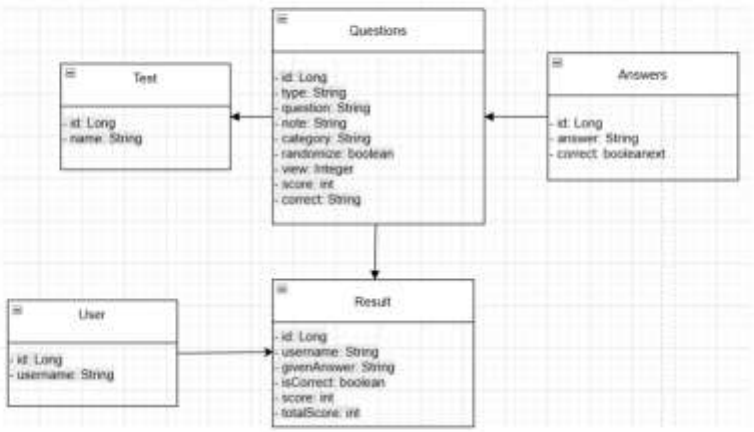
Загальна схема Бази даних додатку



Діаграма використання додатку

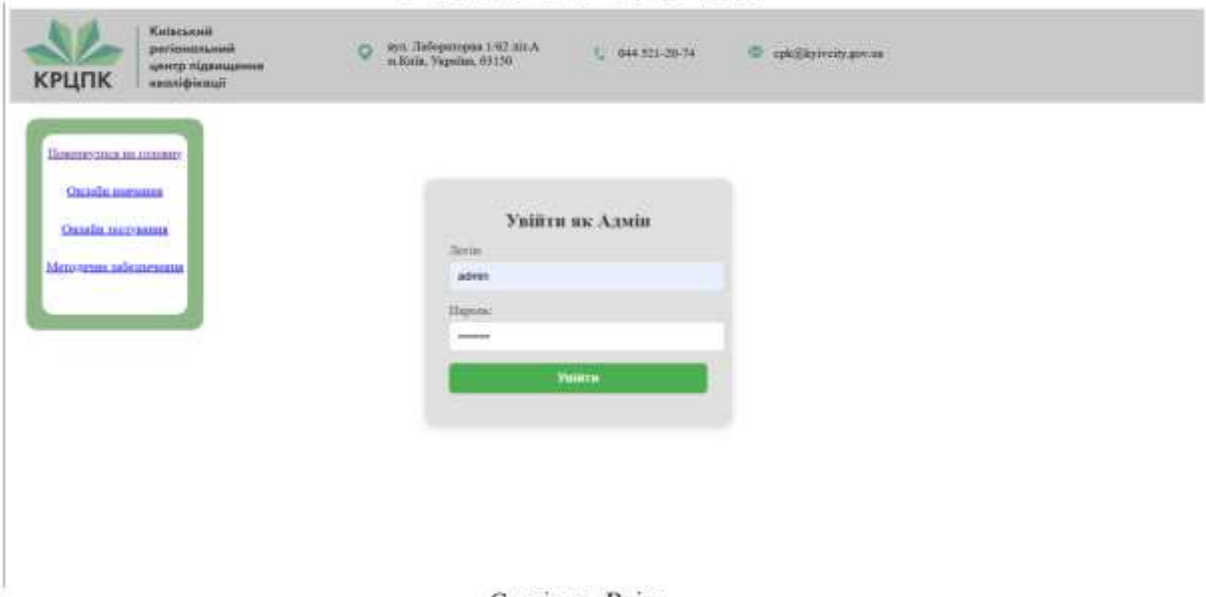


Діаграма класів



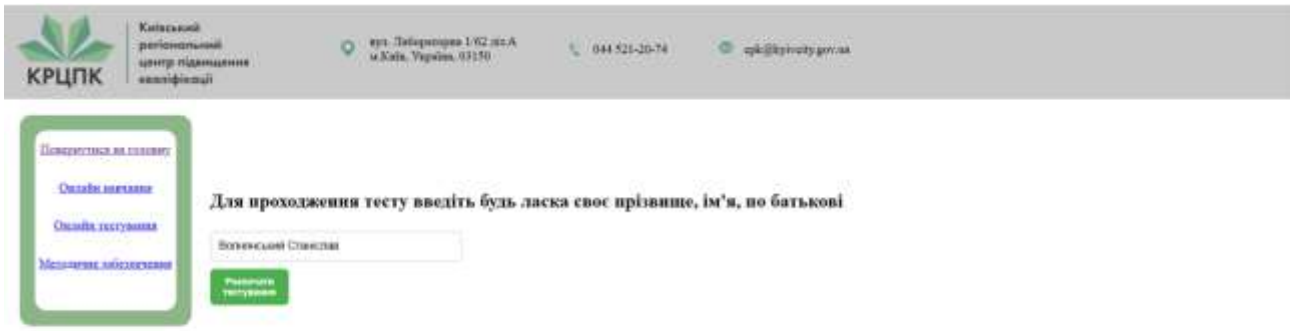
9

ЕКРАННІ ФОРМИ



Сторінка «Вхід»

10



Сторінка «Головна»

11

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Волнянський С.Р., Яскевич В.О Визначення вимог до сайту тестування публічних службовців. Матеріали:VI Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». 25.04.2025, ДУІКТ, Київ, Україна, С. (подано до друку)
2. Волнянський С.Р., Яскевич В.О. Переваги використання Spring Boot у розробці вебзастосунку для онлайн-тестування публічних службовців. Матеріали:VI Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». 25.04.2025, ДУІКТ, Київ, Україна, С. (подано до друку)

12

ВИСНОВКИ

1. Вивчення існуючих систем тестування дозволило виявити їх сильні та слабкі сторони. Оцінка їх функціональності, зручності користування та надійності допомогла визначити ключові вимоги до розроблюваної системи. Виявлені недоліки виявились основними орієнтирами для удосконалення користувацького інтерфейсу, полегшення процесу тестування та забезпечення надійності системи.
2. Було створено структуру бази даних для збереження тестів, питань, варіантів відповідей та результатів користувачів. Структура забезпечує підтримку багаторазового проходження тестів, зберігання історії відповідей і результатів, що дає змогу ефективно управляти тестуванням та аналізувати досягнення користувачів.
3. Реалізовано механізм збереження відповідей користувачів та підрахунку результатів з урахуванням правильності відповідей і нарахуванням балів. Це дозволяє точно оцінити результати тестувань та забезпечити коректне нарахування балів, що є важливим для оцінки ефективності навчання.
4. Реалізовано можливість експорту результатів тестування в форматі Excel, що дає змогу адміністраторам зберігати, аналізувати або друкувати результати для подальшого використання, що значно підвищує зручність використання системи в умовах навчального процесу.
5. Впроваджено функціонал входу адміністратора, який дозволяє редагувати існуючі тести та створювати нові. Це забезпечує гнучкість і зручність управління тестами, що є необхідним для забезпечення постійного оновлення і адаптації системи до змінюваних вимог.
6. Реалізовано функціонал для перегляду результатів тестування, включаючи можливість перегляду неправильних відповідей та загального балу. Це дозволяє користувачам детально аналізувати свої помилки та досягнення, що сприяє покращенню результатів наступних тестувань.

ДОДАТОК Б. ЛІСТЕНІНГИ ПРОГРАМНИХ МОДУЛІВ

Вихідний код файлу Model

```
@Entity
@Table(name = "answers", schema = "tests")
public class Answers {
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
 private Long id;
@ManyToOne(fetch = FetchType.LAZY)
@JoinColumn(name = "question_id", nullable = false)
private Questions question;
@Column(nullable = false)
private String answer;
@Column(name = "is_correct", nullable = false)
private boolean correct;
public boolean isCorrect() {
return correct;
}
public void setCorrect(boolean correct) {
this.correct = correct;
}
public Long getId() {
return id;
}
public void setId(Long id) {
this.id = id;
}
public Questions getQuestion() {
return question;
}
public void setQuestion(Questions question) {
this.question = question;
}
public String getAnswer() {
return answer;
}
public void setAnswer(String answer) {
this.answer = answer;
}
}
```

```
@Entity
@Table(name = "results", schema = "tests")
public class Result {
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private Long id;
@Column(nullable = false)
private String username;
@ManyToOne
@JoinColumn(name = "question_id", nullable = false)
private Questions question;
@Column(nullable = false)
private String givenAnswer;
@Column(nullable = false)
private boolean isCorrect;
@Column(nullable = false)
private int score = 0;
@Column(nullable = false)
private int totalScore = 0;
public int getScore() {
return score;
}
public void setScore(int score) {
```

```
this.score = score;
}
public Long getId() {
return id;
}
public void setId(Long id) {
this.id = id;
}
public String getUsername() {
return username;
}
public void setUsername(String username) {
this.username = username;
}
public Questions getQuestion() {
return question;
}public void setQuestion(Questions question) {
this.question = question;
}
public String getGivenAnswer() {
return givenAnswer;
}
public void setGivenAnswer(String givenAnswer) {
this.givenAnswer = givenAnswer;
}
public boolean isCorrect() {
return isCorrect;
}
public void setCorrect(boolean correct) {
isCorrect = correct;
}
public int getTotalScore() {
return totalScore;
}
public void setTotalScore(int totalScore) {
this.totalScore = totalScore;
}
}
```

Вихідний код контролера відповідального за вивід всіх тестів на сторінці

```
@GetMapping("/available-tests")
public String showAvailableTests(Model model) {
List<Test> tests = testRepository.findAll();
model.addAttribute("tests", tests);
return "available-tests";
}
```