МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

_____ проф. Приходько С.Б.

«___» _____ 2022 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: "Розробка веб-застосунку для тестування та діагностики рівня професійних знань здобувачів вищої освіти"

Виконав: студент групи 4151

Петриченко С.А.

(підпис, ПІБ)

Керівник роботи:

Доцент кафедри ПЗАС, к.т.н., доцент

(посада, науковий ступень вчене звання)

Устенко І.В.

(підпис, ПІБ)

Миколаїв – 2022 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)
«___» _____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту Петриченку Сергію Андрійовичу
(Прізвище, ім'я, по батькові)

1. Тема роботи: "Розробка веб-застосунку для тестування та діагностики рівня професійних знань здобувачів вищої освіти". "Web application development for testing and diagnosing the level of professional knowledge of university students"

Керівник роботи Устенко Ірина Валеріївна
Затверджені наказом ректора №256-уч від «11» травня 2022 р.
2. Термін подання роботи: 10.06.2022 р.
3. Вихідні дані по роботі: _____

4. Перелік питань, що належать до розробки (найменування розділів) _____
Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Опис предметної галузі та постановка задачі; Проєкт програмного забезпечення (ескізний, технічний та робочий); Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).

5. Перелік презентаційних матеріалів
Тема кваліфікаційної роботи, Мета кваліфікаційної роботи, Функції програмного забезпечення, Постановка задачі, Діаграми діяльності для процесу "Авторизація", Діаграмма діяльності для процссу "Перегляд загального списку тестів", Діаграма класів, Логічна модель структури бази даних, Діаграма розгортання програмного забезпечення, Результат роботи, Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Гурець Н.В., ст. викладач		

7. Дата видачі завдання 11.05.2022 р.

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу Вступ	24.03.2022	ПП*
2.	Опис та аналіз предметної галузі	28.03.2022	ПП*
3.	Постановка задачі	31.03.2022	ПП*
4.	Ескізний проєкт програмного забезпечення	27.04.2022	
5.	Технічний проєкт програмного забезпечення	06.05.2022	
6.	Робочий проєкт програмного забезпечення	13.05.2022	
7.	Підготовка розділу Результати розробки програмного забезпечення	17.05.2022	
8.	Підготовка розділу з охорони праці	18.05.2022	ПП*
9.	Підготовка розділу Висновки	19.05.2022	
10.	Оформлення списку використаних джерел та додатків	20.05.2022	
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	10.06.2022	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку)
12.	Підготовка презентації та доповіді	12.06.2022	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	14.06.2022	
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді	24.06.2022	

* - за результатами переддипломної практики (ПП), яка була з 21.03.2022 до 08.05.2022 р.

Студент _____ (підпис) Петриченко С.А. (ПІБ)

Керівник роботи _____ (підпис) Устенко І.В. (ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота присвячена розробці програмного забезпечення для автоматизації тестування та діагностики рівня професійних знань серед здобувачів вищої освіти. Метою розробки програмного забезпечення є реалізація веб-додатку який покращить та спростить процес тестування та збір результатів тестів, зменшить потребу участі викладацького складу у процесі оцінювання, вивільнивши додатковий час, який раніше був залучений для перевірки однотипних завдань.

Кваліфікаційна робота у своєму складі містить : етапи розробки програмного забезпечення, цілі та призначення, опис, структуру, основні функції, цілі розробки.

Також представлений опис та аналіз основних сучасних методів проведення оцінки рівня знань. Об’єктно-орієнтована методологія була взята як основна модель при розробці програмного забезпечення.

Робота містить 99 сторінки друкованого тексту, 27 рисунка та список використаної літератури із 7 найменувань.

Робота виконана українською мовою.

ABSTRACT

Thesis is devoted to the development of software for automation of testing and diagnostics of the level of professional knowledge among graduates. The purpose of software development is to implement a web application that will increase and simplify the testing process and collection of test results, reduce the need for teaching staff in the assessment process, freeing up additional time previously used to test similar tasks.

Qualification thesis consists of: stages of software development, goals and purposes, description, structure, main functions, development goals.

There is also a description and analysis of the main modern methods of assessing the level of knowledge. Object-oriented methodology has been used as the main model in software development.

The work contains 99 pages of printed text, 27 figures and a list of references from 7 titles.

The work is done in Ukrainian.

Зміст

ВСТУП

В сучасному світі значну частину життя будь-якої людини займає навчання. Отримання нових навичок, вдосконалення та поглиблення існуючих – це основа успіху у сьогоденні. З розвитком освітнього процесу сформувались певні системи оцінки знань, це дало можливість провести стандартизацію та уніфікацію дипломів, сертифікатів, навчальних програм.

Один з найвідоміших методів оцінки навчальних успіхів – це тестування. Тест – це методологія виявлення професійних якостей особистості та рівень їх вираження, обсягу та якості знань, навичок та вмінь людини за допомогою спеціально підготовлених задач, питань, ситуацій і т.д. У практиці тестування-діагностики розрізняють психологічні та педагогічні тести. Педагогічний тест – це система завдань специфічної форми, визначеного змісту, зростаючої трудності, яка дозволяє якісно оцінити структуру та виміряти рівень знань, умінь та навичок. Сучасний тест – це обмежений за часом та стандартизований іспит, що дозволяє зі значною впевненістю у результаті визначити актуальний рівень знань. Для того щоб бути впевненими у результаті, до кожного тесту висуваються певні критерії, такі як:

- Коректність та однозначність;
- повнота;
- відсутність суперечностей;
- упорядкованість за важливістю;
- можливість перевірити результат у вільних джерелах;
- можливість покращувати та змінювати тестову базу.

Кожен функціональний тест повинен мати такі елементи:

- Проста та зрозуміла інструкція для роботи;
- основна частина (питання або завдання);
- варіанти відповідей;
- оцінка.

- В результаті завданням кваліфікаційної роботи було проаналізувати процес проходження тестів та отримання кінцевих результатів, а також сформулювати постановку задачі, проаналізувати наявних рішення на доступному ринку програмного забезпечення, перевірити доцільність розробки власного програмного забезпечення у вигляді веб-застосунку, побудувати інформаційну модель, модель варіантів використання та реалізувати ПЗ.

Існують веб-застосунки для розв’язання цієї задачі, наприклад: Google Forms та tipeform. Але вони не повній мірі враховують специфіку педагогічної діяльності та мають певні архітектурні обмеження. Тому було прийнято рішення про створення власного програмного продукту у вигляді веб-застосунку.

Метою розробки веб-застосунку для тестування та діагностики рівня професійних знань є автоматизація процесів створення та проходження тестів, отримання готових результатів та статистики студентом та викладачем, реалізувати задачу інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов.

Для розробки програмного забезпечення необхідно розв’язати наступні задачі:

- провести аналіз існуючих систем тестування, діагностики;
- провести аналіз процесів формування, збереження та передачі тестових завдань та відповідей та питання;
- провести аналіз існуючих рішень для автоматизації процесу тестування;
- виконати постановку задачі;
- розробити ескізний та технічний проекти ПЗ;
- реалізувати ПЗ;
- ознайомитися з правилами техніки безпеки та охорони праці при роботі з комп’ютером.

1 АНАЛІЗ ПРОЦЕСУ ТЕСТУВАННЯ РІВНЯ ЗНАНЬ. ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Аналіз методів та процесу тестування

Контроль знань студенів з використанням тестів має ряд переваг в порівнянні з традиційними способами контролю знань:

- Тестування універсальний інструмент. Тест може містити в себе завдання, з усіх тем курсу. Це дозволяє оцінити знання студента по всьому курсу, а не за декількома темами, що потрапили в квитку;
- тест є більш точним інструментом, так, наприклад, шкала оцінювання тесту залежить від кількості питань. Чим більша кількість питань, тим більш поглиблений зріз знань можливо провести;
- тестування більш ефективне з економічного погляду. Основні витрати припадають на розробку тесту, тобто мають разовий характер. Витрати ж на проведення тесту значно нижче, ніж за письмового чи усного контролю;
- Тестування значно скорочує час проведення контролю знань (на отримання результату діагностики витрачається приблизно вчетверо менше часу, ніж на проведення традиційного іспиту). Використання обчислювальної техніки та програмних засобів у проведенні тестового контролю знань ще більше підвищило ефективність тестування. Комп'ютерне тестування знань дозволило отримати оцінку рівня навчальних досягнень незалежну від суб'єктивної думки екзаменатора. Однак для забезпечення об'єктивності оцінки знань та ефективності комп'ютерного тесту необхідно успішно розв’язати дві суттєві проблеми:
    - розробити якісний тестовий матеріал;

- розробити інструментарій тестової системи та вжити спеціальних заходів щодо забезпечення конфіденційності тестових завдань[1].

Питання інструментарію та конфіденційності комп'ютерного тестування вирішується наявністю готових програмних рішень. Але розробка якісних контролюючих матеріалів та побудова адекватної моделі тестування вважається серйозним завданням. Саме від повноти та різноманітності створених контролюючих матеріалів, способів їх використання, здатності створеної системи гнучко реагувати на дії учня залежить ефективність комп'ютерного контролю та об'єктивність оцінки піддослідних. Побудувати досить ефективну модель діагностики можна за допомогою адаптивного тесту. Такий тест являє собою варіант системи тестування, якій заздалегідь відомі параметри важкості та диференційна здатність кожного завдання. Сама система має вигляд комп'ютерного банку завдань з усіх тематичним модулям, упорядкованим відповідно до рівня їх труднощі.

Важкість завдань визначається досвідченим шляхом: перш ніж потрапити до банку, кожне завдання проходить емпіричну апробацію на досить великій кількості типових учнів цікавого контингенту. Вихідний тест кожного модуля попередньо тестується на групі учнів з метою наближеного визначення середнього рівня навченості групи з відповідного модуля курсу. Ефективність адаптивного тесту та достовірність оцінки результатів навчання, забезпечується за рахунок оптимізації процедур вибору, видачі та оцінки результатів виконання адаптивних тестів При видачі завдань використовується багатокрокова стратегія: черговий крок відбувається лише після оцінки результатів виконання попереднього кроку.

Якщо тестований успішно відповів питанням тесту, робиться висновок, що рівень його загальної підготовки вищий за складність пред'явленого завдання та йому надається тестове завдання більшої складності з того ж тематичного модуля. Якщо тестований не відповів на запитання тесту, то йому буде запропоновано ще одна спроба вирішення завдання тієї ж

проблеми. Якщо вона також не вирішена, то Пред'являється завдання зниженої складності. Якщо одразу не вирішено менш важке завдання, то пропонується завдання ще меншої складності. В результаті, випробуваному буде обрано певний рівень труднощі, навколо якого і буде розподілятися труднощі завдань, що видаються. В теперішній час виділяють три варіанти адаптивного контролю[7]:

- такі як пірамідальне тестування;
- flexi level-тестування;
- stradaptive-тестування.

При пірамідальному тестуванні на першому етапі всім випробуваним видаються завдання однакового середнього рівня складності, що визначається як середнє між найнижчим та найвищим рівнем. Якщо відповідь на запитання середньої Проблеми неправильний, то проблема наступного питання буде визначатися як середнє між найнижчим рівнем труднощі та тим середнім, на який він не відповів. Т.ч. відбувається постійне розподіл шкали проблеми завдань навпіл. При flexi level-тестуванні на першому етапі завдання будь-якого (не середнього) рівня складності. При stradaptive-тестуванні кожне наступне завдання відрізняється за складністю від попереднього на один рівень.

Для математичної формалізації завдання адаптивного контролю використовуються два підходи: теоретичні основи моделі Раша або основи теорії моделювання та параметризації Item Response Theory (IRT). При статистичній обробці відповідей адаптивного тестування перша використовує апарат нечітких множин, а друга для моделювання ймовірностей правильних відповідей логістичну криву Доцільність адаптивного контролю знань визначається оптимізацією процесу тестування, тому що немає необхідності давати легкі завдання знаючому студенту та складні завдання мало підготовленому. Використання завдань, відповідних рівню підготовки, дозволяє зменшити час тестування та підвищити точність виміру рівня знань. Адаптивне тестування направлено на адаптацію процесу навчання до

індивідуальних особливостей в умовах колективного навчання, безпосередньо з адаптивним тестуванням пов'язане поняття адаптивного навчання, яке дозволяє забезпечити подання навчального матеріалу на оптимальному 50% рівні складності, оскільки легкі завдання не мають потенціал, а дуже складні знижують мотивацію до навчання

1.2 Недоліки існуючих систем оцінки рівня професійних знань

Під час аналізу існуючих систем оцінки рівня професійних знань було виявлено, що стандартна система тестування пов`язана з накопиченням та оборотом великої кількості інформації на традиційних паперових носіях, що в результаті ускладнює процес тестування та аналізу результатів, а також потребує значної кількості часу для перевірки результатів опитувань.

В процесі створення тесту для оцінки рівня знань зазвичай дотримуються загальноприйнятого формату тесту, який передбачає наявності роздрукованого аркушу з питаннями та варіантами можливих відповідей та номером варіанту, на другому аркуші для відповідей повинна бути інформації про навчальний заклад в якому проходить тестування, предмет тестування, тему, поля для запису індивідуальних даних студента, такі як учбова група, спеціальність, дату проведення тестування та поле для запису кінцевого результату.

Усі аркуші по завершенню тестування повертаються викладачу. При цьому кожного разу витрачається певний час, для того, щоб систематизувати та скласти матеріали відповідей студентів. Викладач має зберігати значну кількість паперових матеріалів з тестовими завдання для кожної теми навчального предмета, а також всі письмові результати тестувань усіх студентів навчального курсу. Це може створити труднощі, пов’язані зі способом зберігання паперових бланків, а також може призвести до втрати бланків із результатами тестування через людський фактор.

Питання опитувань та відповіді також створюється і

зберігається викладачем у вигляді паперових матеріалів. Стандартний тест містить інформацію про тему тестування, перелік питань систематизованих по підтемах, зі збільшенням рівня важкості. Отже, існуюча система організації введення та накопичення даних має наступні недоліки:

- Готовий тест у паперовому вигляді неможливо редагувати або видалити зафіксовану інформацію. Для того, щоб змінити інформацію, необхідно заново роздруковувати новий варіант;
- при великій кількості студентів та тестів потрібно накопичувати велику кількість паперових бланків, а також витрачати значний час на організацію, пошук та зберігання інформації.

1.3 Аналіз існуючих програмних продуктів, їх недоліків та переваг

Розглянемо актуальні веб-застосунки для автоматизації проведення тестування у педагогічній практиці, проаналізуємо їх переваги та недоліки.

Google Forms – це веб-застосунок для автоматизації тестів, а також опитувань. Застосунок працює в хмарі Google, зібрані дані зберігаються у таблицях Google тому вся статистична інформація доступна в будь-якій точці світу при наявності підключенні до інтернету[4].

Інтерфейс веб-застосунку «Google Forms» представлено на рис. 1.1.

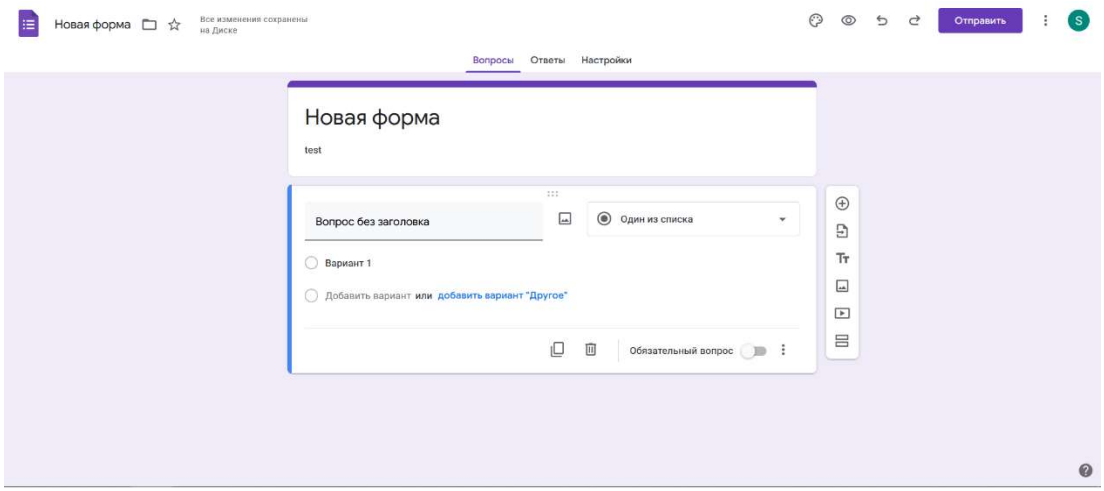

Рисунок 1.1 – Інтерфейс веб-застосунку «Google Forms»

Веб-застосунок вирішує наступні завдання:

- формування тестових форм, яке включає можливість додавати/видаляти питання, включати у окремі питання окремі розділи, а також фото та відео матеріали;
- створення єдиного робочого простору між різними користувачами;
- перегляд та аналіз результатів тестування різних користувачів;
- синхронізація між різними сервісами Google, такі як: Таблиці, Презентації, Документи, Форми;
- можливість створювати шаблони друку з Google Forms для перенесення тестів у паперовий вигляд;
- можливість переходу на бізнес-версії для розширення функціональних можливостей.

Typeform – це веб-застосунок для автоматизації створення багатофункціональних форм. Дане рішення орієнтоване на складних форм, та складних систем зі значним навантаженням.

Інтерфейс веб-застосунку «Typeform» представлено на рис. 1.2.

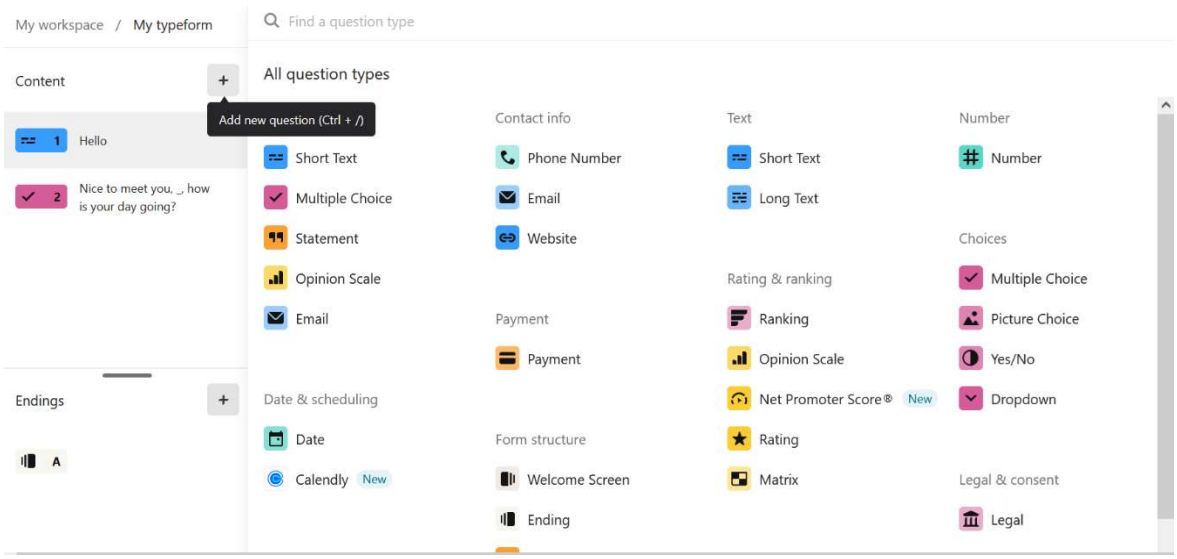

Рисунок 1.2 – Інтерфейс веб-застосунку «Typeform»

Веб- застосунок вирішує наступні завдання:

- формування тестових завдань;
- включення до форм додаткових посилань, особистих нотаток;
- можливість включати власні файли для перевірки у якості відповіді;
- створення календаря тестування з системою оповіщень;
- можливість формувати нові тест з готових варіантів з різними конфігураціями(кількість питань, складність і т. д.);
- можливість інтеграції аккаунт з Slack, Google Таблиці.

Недоліки перелічених веб-застосунків:

− обмеженість безкоштовних версій;

− загальний напрямок застосування, не відповідність унікальним потребам навчальних закладів;

− дані користувачів зберігаються на віддалених серверах комерційних компаній.

У ході аналізу існуючих рішень не було виявлено веб-застосунку для автоматизації тестування, який міг би відповідати всім вимогам навчального закладу.

1.4 Постановка задачі на розробку програмного забезпечення для автоматизації процесу тестування та діагностики рівня професійних знань серед здобувачів вищої освіти

Аналізуючи предметну галузь та наявні у вільному доступі найбільш популярні програмні засоби була сформульована наступна задача – розробити програмне забезпечення у вигляді веб-застосунку для автоматизації процесу проходження навчальних тестів. Метою розробки ПЗ є полегшення проходження тестувань серед студентів та систематизація отриманих результатів для подальшого аналізу успіхів.

Провівши аналіз предметної галузі та аналіз наявних програмних засобів можливо зробити висновок, що існуючі ПЗ мають певні недоліки, не дозволяють мати повний контроль за програмним засобом та базами даних . В результаті аналізу було прийнято рішення розробити власну платформу для внутрішнього застування у Закладах вищої освіти.

1) Вимоги до функціональних характеристик

Програмне забезпечення у вигляді веб-застосунку повинно мати авторизацію для користувачів та окремі ролі: студент, адміністратор.

Для адміністратора мають бути реалізовані наступні функції:

- Створення нових тестів;
- редагування тестів;
- видалення тестів;
- створення нових запитань;
- редагування запитань;
- видалення запитань;
- створення відповідей на запитання;
- видалення відповідей на запитання;
- редагування відповідей на запитання;
- перегляд користувачів;
- перегляд результатів тестування користувачів.

Для користувача мають бути реалізовані наступні функції:

- Перегляд доступних тестів;
- проходження доступних тестів;
- перегляд результатів тестів;
- перегляд загальної статистики тестування.

2) Вимоги до організації вхідних та вихідних даних

Введення даних має здійснюватися користувачем за допомогою персонального комп`ютера або смартфону. Усі дані зберігаються на сервері.

Отримана інформація виводиться на екрані у вигляді HTML-сторінки.

Вхідні дані:

- Інформація про обліковий запис;
- інформація про тести;
- інформація про питання;
- інформація про відповіді.

Вихідні дані:

- Список тестів;
- список питань;
- список відповідей;
- список результатів;

3) Вимоги до складу та параметрів технічних засобів

Програмне забезпечення можливо використовувати н будь-якій обчислювальній техніці, що має браузер загального використання(Опера, Гугл браузер).

Середні вимоги для браузера

- Пам’ять: 512 Mb;
- здатність екрана > 800 х 600 пікселів.

4) Вимоги до інформаційної та програмної сумісності

Веб-застосунок пред'являє лише наступні вимоги до обчислювальної техніки:

- Операційна система: Android 5.0+, Windows XP+;
- Доступ в мережу Інтернет.

Вимоги до розробки програмного забезпечення наведені у додатку А.

2 ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСУ ТЕСТУВАННЯ ТА ДІАГНОСТИКИ РІВНЯ ПРОФЕСІЙНИХ ЗНАНЬ СЕРЕД ЗДОБУВАЧІВ ВИЩОЇ ОСВІТИ

2.1 Ескізний проект програмного забезпечення для автоматизації процесу тестування рівня професійних знань серед здобувачів вищої освіти

У проекті представлена основна проектна документація, а також опис конструктивних рішень, що були підготовлені у результаті аналізу вимог до майбутнього програмного забезпечення та дає розуміння про те, як буде використовуватись програмне забезпечення, у вигляді веб-застосунку для автоматизації процесу тестування рівня професійних знань серед здобувачів вищої професійної освіти. Методологією для розробки була обрана об’єктно-орієнтована модель. Ескізний проект розроблено з використанням UML-моделювання, а також діаграмами варіантів використання.

2.1.1 Модель варіантів використання

Для опису вимог до майбутнього програмного забезпечення було розроблено діаграму варіантів використання, можливих акторів, зв’язки між ними.

Діаграма варіантів використання дає можливість проаналізувати структурні зв`язки програмного забезпечення, основні функції, методи взаємодії з зовнішніми сутностями.

У результаті аналізу вимог було розроблено варіанти використання:

- Перегляд тестів;
- створення тестів;
- проходження тестів;
- перегляд питань до тестів;
- створення питань;
- створення відповідей до питань;
- вибір правильних відповідей;
- перегляд результатів тестів;

Перелік акторів для програмного забезпечення:

- Студент – основний користувач програмного забезпечення, який проходить тести;
- Адміністратор – створює тести, можливі відповіді, уточнює правильні та не правильні відповіді до питань.

Діаграму варіантів використання зображено на рис. 2.1

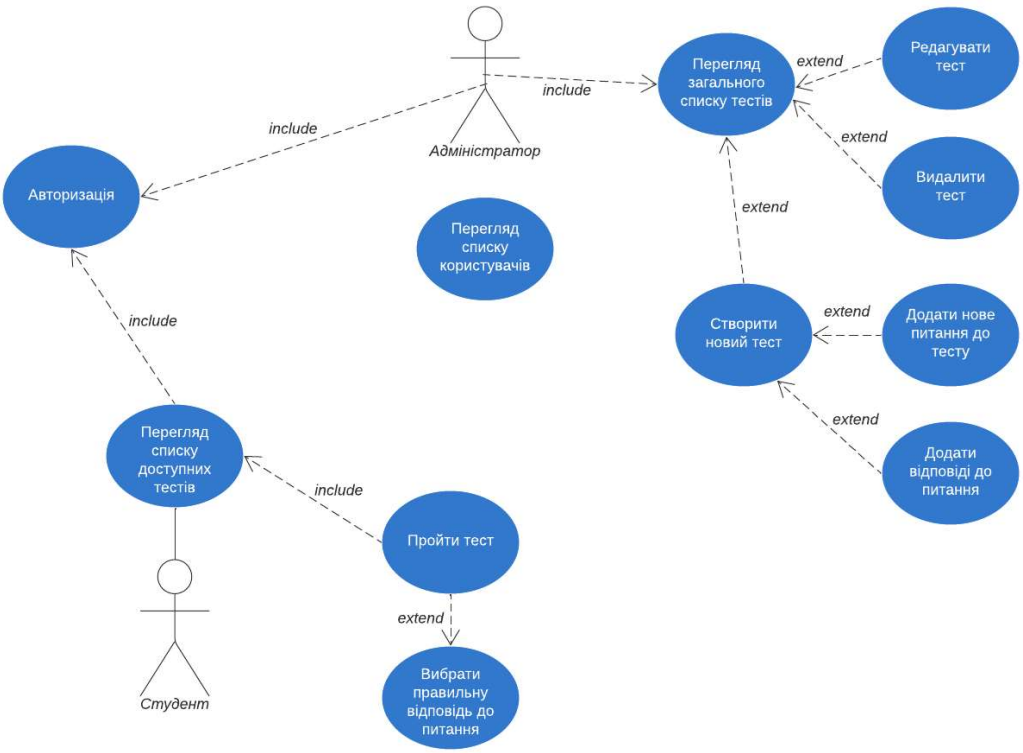

Рисунок 2.1 – Діаграма варіантів використання

Отже, для опису функціональних вимог до програмного забезпечення було побудовано діаграму варіантів використання.

2.1.2 Специфікації варіантів використання

У результаті аналізу діаграми варіантів використання створимо специфікації для кожного з них. Нижче наведені специфікації для кожного з варіантів використання.

Специфікація варіанту використання «Перегляд списку доступних тестів»

1) Найменування: «Перегляд списку доступних тестів»
2) Призначення: для перегляду списку доступних тестів.
3) Учасники: активний суб’єкт «Студент».
4) Передумови: перед активізацією даного варіанта використання повинен бути виконаний потік варіанта використання «Авторизація».
5) Основний потік: Студент після проходження авторизації потрапляє на сторінку тестів.
6) Альтернативні потоки: немає.
7) Спеціальні вимоги: немає.
8) Постумови: виводиться список доступних тестів.
9) Додаткові зауваження: додаткових зауважень немає.

Специфікація варіанту використання «Пройти тест»

1) Найменування: «Пройти тест»
2) Призначення: призначений для проходження тесту.
3) Учасники: активний суб’єкт «Студент».

4) Передумови: перед активізацією даного варіанта використання повинен бути виконаний потік варіанта використання «Перегляд списку доступних тестів».
5) Основний потік: Студент після проходження авторизації та перегляду доступних тестів активує сторінку проходження тесту.
6) Альтернативні потоки: немає.
7) Спеціальні вимоги: немає.
8) Постумови: вибрати один з доступних у переліку тестів для проходження.
9) Додаткові зауваження: додаткових зауважень немає.

Специфікація варіанту використання «Авторизація»

1) Найменування: «Авторизація»
2) Призначення: вхід до веб-застосунку, ідентифікація згідно з доступними ролями.
3) Учасники: активний суб’єкт «Студент», активний суб’єкт «Адміністратор».
4) Передумови: перейти на адресу веб-застосунку використовуючи доступний браузер.
5) Основний потік: необхідно ввести індивідуальний логін та пароль у системі. Порівнюється логін та пароль з наявними логінами у базі даних користувачів, а також перевіряється відповідність паролю, у випадку коли логін користувача присутній у відповідній таблиці бази даних. У випадку коли логін не знайдено – активується альтернативний потік.
6) Альтернативні потоки: відображається повідомлення про помилку.
7) Спеціальні вимоги: не визначені.
8) Постумови: розподіл прав доступу.

9) Додаткові зауваження: немає.

Специфікація варіанту використання «Перегляд списку користувачів»

1) Найменування: «Перегляд списку користувачів»
2) Призначення: для перегляду списку зареєстрованих користувачів.
3) Учасники: активний суб’єкт «Адміністратор».
4) Передумови: перед активізацією даного варіанта використання повинен бути виконаний потік варіанта використання «Авторизація».
5) Основний потік: Після проходження авторизації потрапляє на головну сторінку та переходить на сторінку списку користувачів.
6) Альтернативні потоки: немає.
7) Спеціальні вимоги: немає.
8) Постумови: немає.
9) Додаткові зауваження: додаткових зауважень немає.

Специфікація варіанту використання «Перегляд загального списку тестів»

1) Найменування: «Перегляд загального списку тестів»
2) Призначення: призначений для перегляду списку зареєстрованих тестів доступних для студентів.
3) Учасники: активний суб’єкт «Адміністратор».
4) Передумови: перед активізацією даного варіанта використання повинен бути виконаний потік варіанта використання «Авторизація».
5) Основний потік: Після проходження авторизації потрапляє на головну сторінку та переходить на сторінку списку тестів.
6) Альтернативні потоки: немає.

7) Спеціальні вимоги: немає.
8) Постумови: Виводить список тестів.
9) Додаткові зауваження: додаткових зауважень немає.

Специфікація варіанту використання «Редагувати тест»

1) Найменування: «Редагувати тест»
2) Призначення: редагування складу тестових завдань.
3) Учасники: активний суб’єкт «Адміністратор».
4) Передумови: перед активізацією варіанта використання повинен бути виконаний потік варіанта використання «Перегляд загального списку тестів».
5) Основний потік: Після перегляду списку тестів переходить на сторінку редагування обраного тесту.
6) Альтернативні потоки: немає.
7) Спеціальні вимоги: немає.
8) Постумови: змінюється склад обраного тесту.
9) Додаткові зауваження: немає.

Специфікація варіанту використання «Створити новий тест»

1) Найменування: «Створити новий тест»
2) Призначення: створення нового тесту.
3) Учасники: активний суб’єкт «Адміністратор».
4) Передумови: перед активізацією варіанта використання повинен бути виконаний потік варіанта використання «Перегляд загального списку тестів».
5) Основний потік: Після перегляду списку тестів переходить на сторінку створення нового тесту.
6) Альтернативні потоки: немає.
7) Спеціальні вимоги: немає.
8) Постумови: формується новий тест.

9) Додаткові зауваження: немає.

2.1.3 Побудова сценаріїв використання

Діаграми діяльності для основних варіантів використання Діаграма діяльності варіанту «Авторизація» представлена на рис. 2.2

Рисунок 2.2 – Діаграма діяльності варіанту «Авторизація»

Користувачу програмного забезпечення необхідно ввести логін та пароль у відповідні поля на сторінці авторизації.

Якщо не всі поля заповнені, користувача з відповідним логіном не знайдено або якщо пароль не вірний – відображається помилка. Якщо всі поля заповнені, та введені коректні дані – користувач програмного забезпечення потрапляє на головну сторінку веб-застосунку.

Діаграма діяльності варіанту «Перегляд загального списку тестів» представлена на рис. 2.3

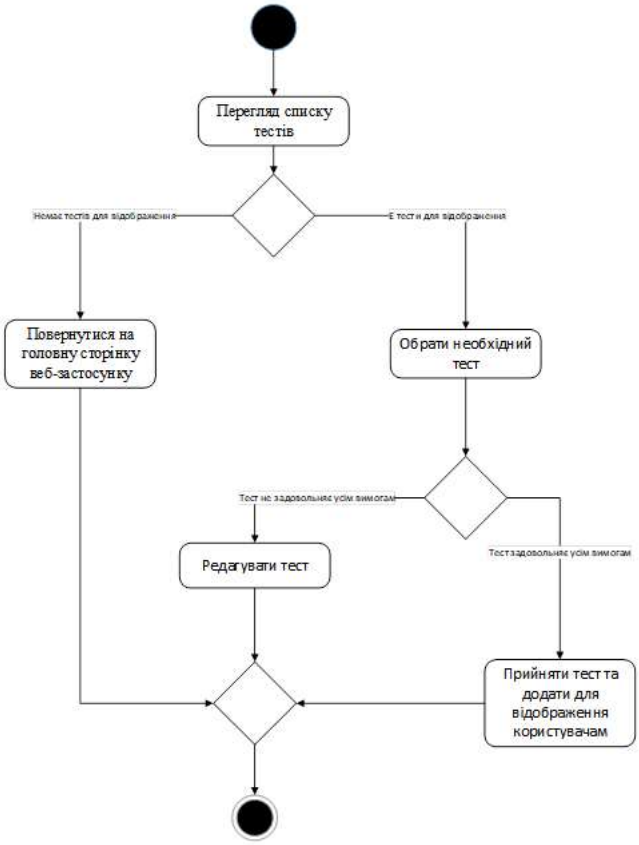

Рисунок 2.3 – Діаграма варіанту використання «Перегляд загального списку тестів»

Адміністратор переходить на сторінку перегляду списку тестів, якщо тестів у базі даних немає - відображається пустий список.

Якщо є тести – вони відображаються у вигляді списку. Обираючи необхідний тест, у випадку якщо тест відповідає усім потребам, його можна відкрити для доступу студентам. В іншому випадку його можливо редагувати для подальшої взаємодії.

Діаграма діяльності для варіанту використання «Створити новий тест» представлена на рис. 2.4.

Рисунок 2.4 – Діаграма варіанту використання «Створити новий тест»

Адміністратор переходить на сторінку перегляду списку тестів, та обирає додати новий тест. Вводить назву тесту та предмет тестування. Якщо усі поля заповнені коректно – відкривається можливість додавати питання. Додавши усі необхідні питання – адміністратор зберігає тест.

2.1.4 Побудова інформаційної моделі

Інформаційна модель програмного забезпечення для автоматизації процесу тестування рівня професійних знань серед здобувачів вищої освіти відображається у вигляді діаграми класів, що базується на діаграмі варіантів

використання і створених діаграмах діяльності. Діаграма класів представлена на рис. 2.5.

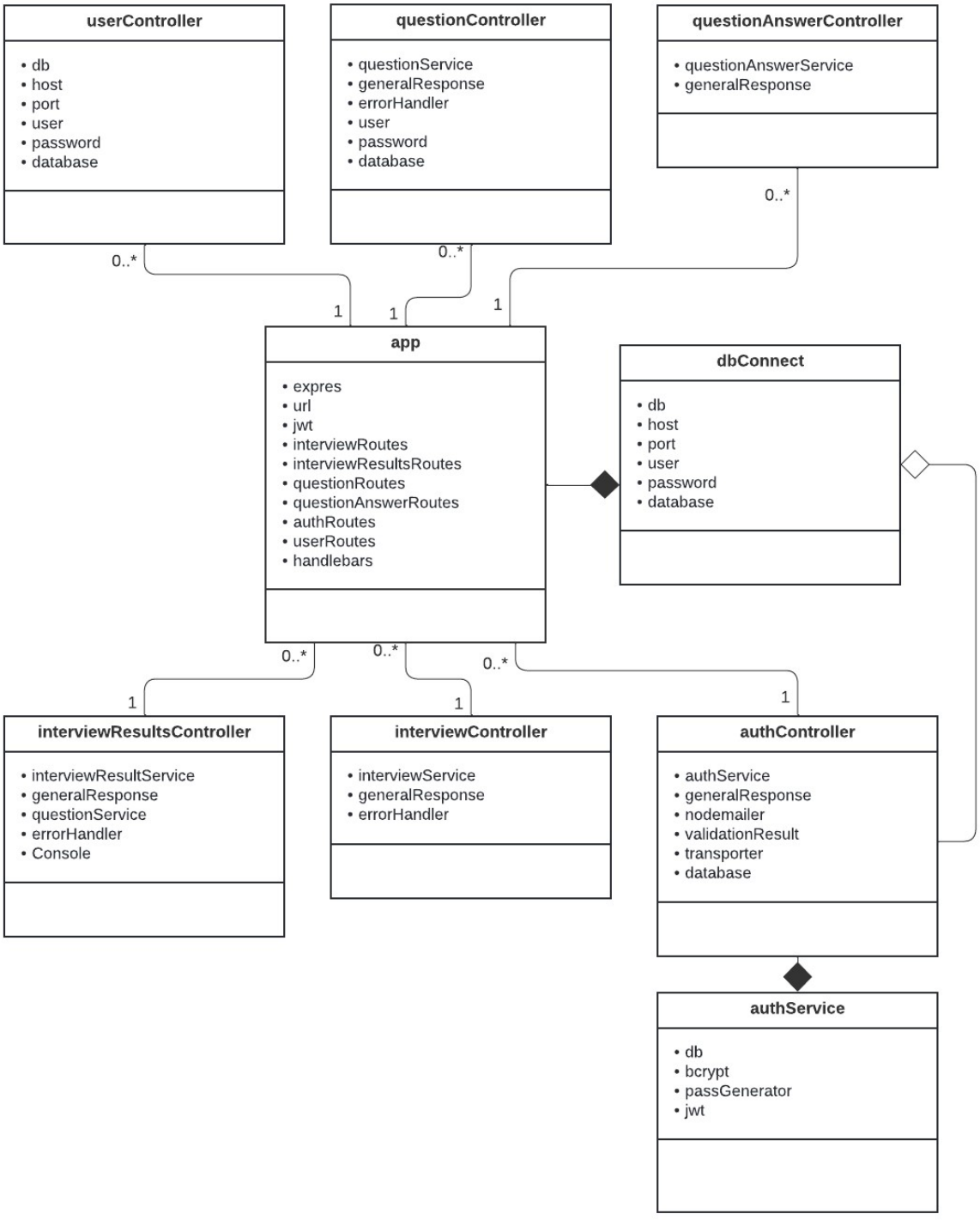

Рисунок 2.5 – Інформаційна модель програмного забезпечення для автоматизації та діагностики процесу тестування рівня професійних знань серед здобувачів вищої освіти.

У результаті було розроблено інформаційну модель програмного забезпечення у вигляді діаграми класів

2.1.5 Розробка інтерфейсу

У підрозділі представлені схеми форм, меню та діалогів для проектування зовнішнього інтерфейсу. Інтерфейс – це засіб зручного відображення та взаємодії користувачів з веб-застосунком.

Інструкція взаємодії користувача з готовим програмним інтерфейсом наведено у Додатку Г.

При відкритті веб-застосунку користувач переходить на сторінку авторизації. Сторінка авторизації представлена на рис. 2.6.

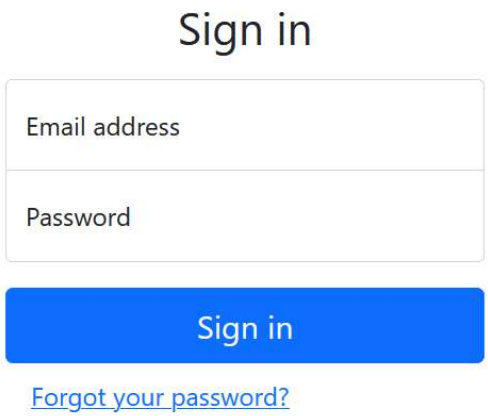

Рисунок 2.6 – Сторінка авторизації при запуску веб-застосунку

При авторизації в систему з правами доступу «Студент», користувач переходить на головну сторінку веб-застосунку студента. При авторизації в систему з правами доступу «Адміністратор», користувач переходить на головну сторінку адміністратора.

Сторінка відновлення паролю зображено на рис. 2.7.

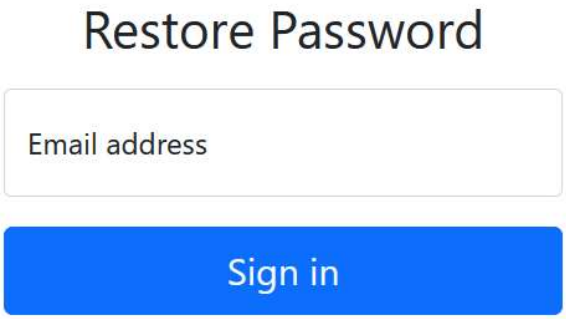

Рисунок 2.7 – Сторінка відновлення паролю

Після натискання на посилання “Forgot your password?”, на сторінці Авторизації відкривається вікно відновлення пароля, в відповідне поле необхідно вести пошту користувача та підтвердити відновлення. Після цього, якщо у базі знайдено користувача з такою поштою, йому буде відправлене посилання для відновлення пароля.

Головну сторінку адміністратора зображено на рис. 2.8.

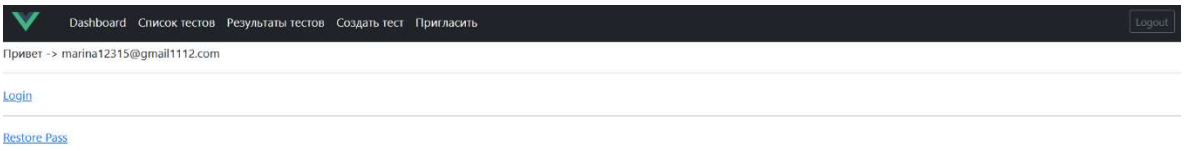

Рисунок 2.8 – Головна сторінка адміністратора

На головній сторінці відображається посилання для відновлення паролю, у шапці - посилання для взаємодії з тестами та студентами, а також кнопка виходу з веб-застосунку.

Головну сторінку студента зображено на рис. 2.9.

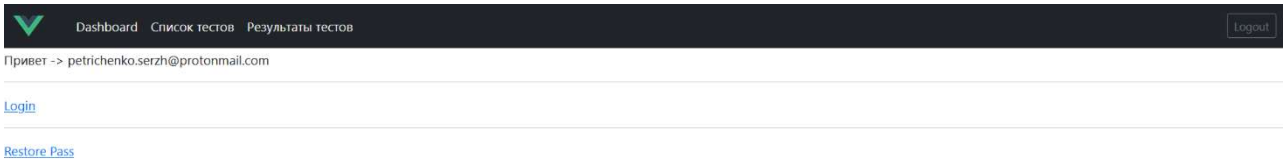

Рисунок 2.9 – Головна сторінка студента

Сторінка зі списком тестів адміністратора зображено на рис. 2.10.

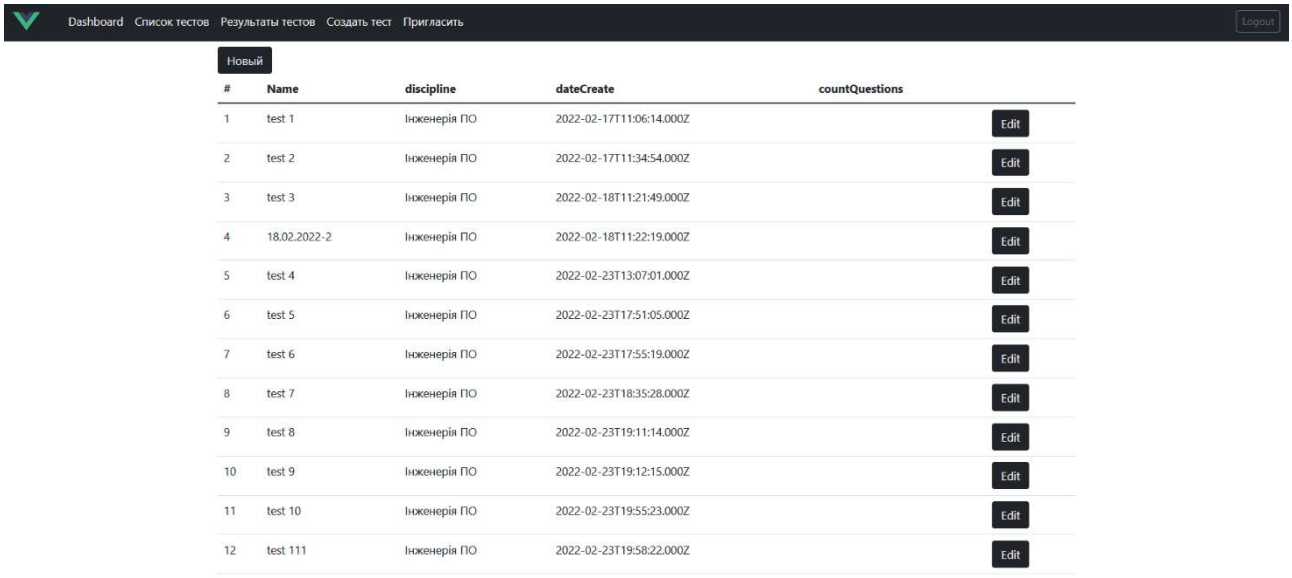

Рисунок 2. 10 – Сторінка зі списком тестів адміністратора

На сторінці відображається список наявних тестів, з назвою конкретного тесту, предмет тестування, а також датою створення тесту. Для адміністратора відкрита можливість редагувати тест, використавши “Edit”

Сторінка редакції тесту зображено на рис. 2.11.

Рисунок 2. 11 – Сторінка редакції тесту

На сторінці відображається список наявних тестів, з назвою конкретного тесту, предмет тестування, а також датою створення тесту. Для адміністратора відкрита можливість редагувати тест, використавши “Edit”

Сторінка створення тесту зображено на рис. 2.12.

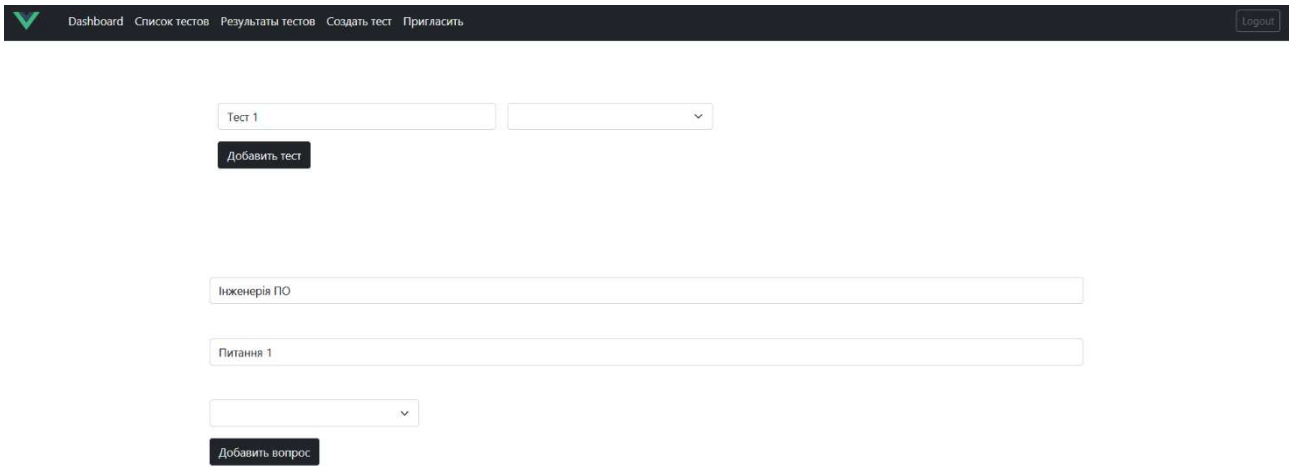

Рисунок 2. 12 – Сторінка створення тесту

На сторінці відображається поля для вводу назви тесту, предмету А також блок додавання питань і відповідей.

2.2 Технічний проект програмного забезпечення для автоматизації процессу тестування та діагностики рівня професійних знань здобувачів вищої освіти.

У результаті аналізу ескізного проекту розроблено технічний проект програмного забезпечення для автоматизації процесу тестування та діагностики рівня професійних знань серед здобувачів вищої професійної освіти шляхом побудови статичних і динамічних моделей. Статична модель представлена діаграмами класів, а динамічна – діаграмами взаємодії.

2.2.1 Розробка структури класів

Використовуючи інформаційну модель, модель варіантів використання, моделі інтерфейсу програмного забезпечення розроблена статична модель, представлена діаграмою класів. Діаграма класів зображена на рис. 2.13.

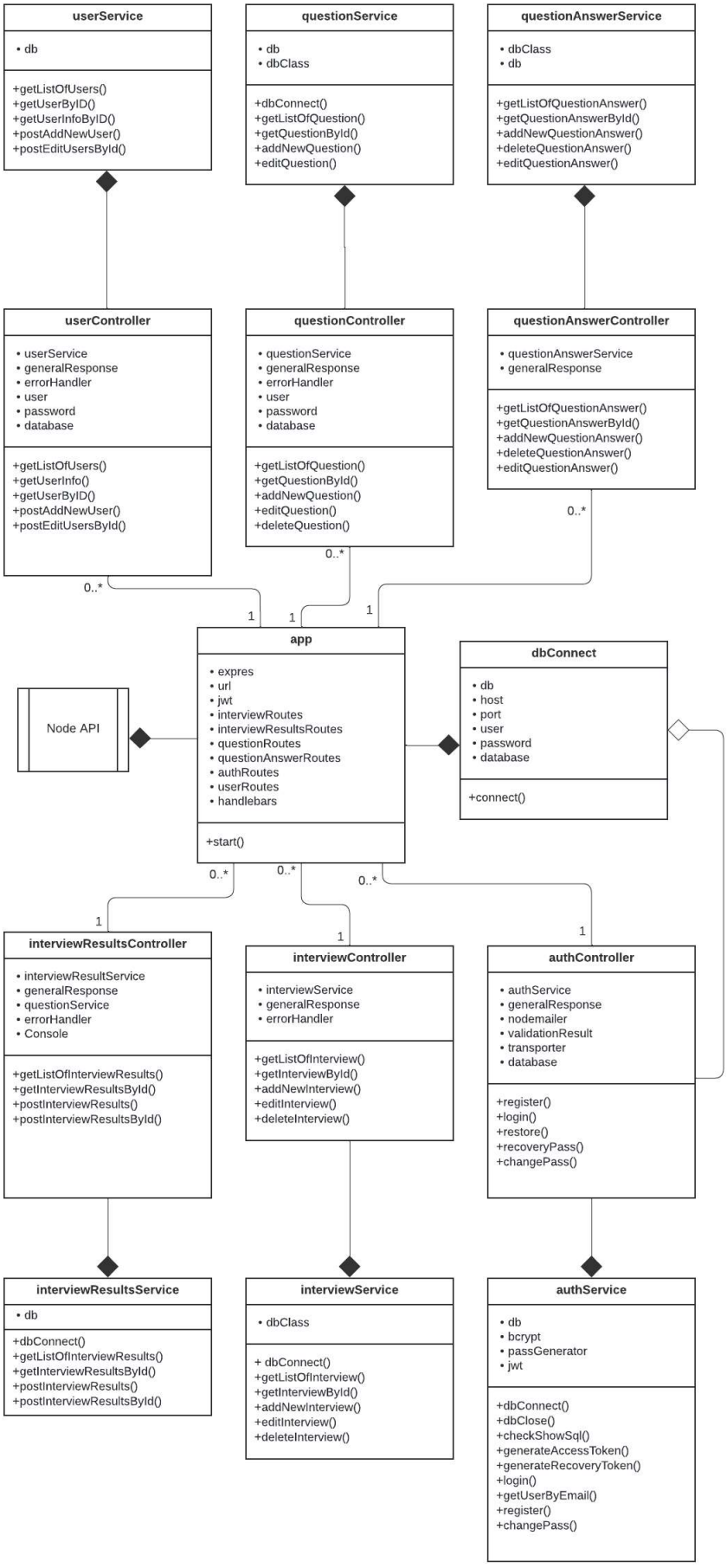

Рисунок 2.13 – Діаграма класів

У результаті аналізу ескізного проекту було розроблено статичну модель, представлену діаграмою класів.

2.2.2 Аналіз специфікації класів

Нижче наведена специфікація класів програмного забезпечення для автоматизації тестування та діагностики рівня професійних знань здобувачів вищої освіти.

Лістинг класів програми наведено у додатку Б.

1) Імя: dbConnect

2) Відповідальність: відповідний за зв'язок з сервером бази даних.

3) Атрибути: хост підключення, порт, логін користувача БД, пароль користувача.

4) Операції:

- connect ()– виконує підключення до бази данних,

1) Ім'я: questionAnswerController

2) Відповідальність: контролює отримання, зміну, відправку, видалення інформації про відповіді на питання тестів.

3) Атрибути: екземпляр класу, шаблон відповіді на запит.

4) Операції:

- getListOfQuestionAnswer() – виконує запит про весь список відповідей на питання тесту;
- getQuestionAnswerById () – виконує запит про весь список відповідей на конкретне питання тесту;
- addNewQuestionAnswer () – додає нову відповідь на запитання тесту;

- deleteQuestionAnswer () – видаляє відповідь на запитання тесту;
- editQuestionAnswer () – змінює відповідь на запитання тесту.

1) Ім’я: questionController
2) Відповідальність: контролює отримання, зміну, відправку, видалення інформації про питання тестів.
3) Атрибути: екземпляр класу, шаблон відповіді на запит, інформація про користувача, базу даних, шаблон помилок.
4) Операції:

- getListOfQuestion () – виконує запит на отримання повного списку питань тесту;
- getQuestionById () – виконує запит на отримання конкретного питання;
- addNewQuestion () – додає нове питання тесту;
- editQuestion () – виконує запит на зміну питання тесту;
- deleteQuestion () – видаляє питання тесту.

1) Ім’я: userController
2) Відповідальність: контролює отримання, зміну, відправку, видалення інформації про користувачів.
3) Атрибути: екземпляр класу, шаблон відповіді на запит, інформація про користувача, базу даних, шаблон помилок.
4) Операції:

- getListOfUsers () – виконує запит на отримання повного списку користувачів;
- getUserInfo () – виконує запит на отримання інформації про користувачів;

- getUserByID () – виконує запит на отримання інформації про конкретного користувача;
- postAddNewUser () – виконує запит на додавання нового користувача;
- postEditUsersById () – редагує дані користувача.

1) Ім’я: questionAnswerService
2) Відповідальність: реалізую методи отримання, зміну, відправку, видалення інформації про відповіді на питання.
3) Атрибути: дані БД.
4) Операції:

- getListOfQuestionAnswer () – реалізую метод отримання списку відповідей на запитання з бази даних;
- getQuestionAnswerById () – реалізую метод отримання списку відповідей на запитання з бази даних;
- addNewQuestionAnswer () – реалізую метод який додає нову відповідь на питання у БД;
- deleteQuestionAnswer () – реалізую метод який видаляє відповідь на питання з БД;
- editQuestionAnswer () – реалізую метод який редагує відповідь на питання з БД.

1) Імя: questionService
2) Відповідальність: реалізую методи отримання, зміну, відправку, видалення інформації про питання.
3) Атрибути: дані БД.
4) Операції:

- dbConnect () – реалізую метод взаємодії з БД;

- getListOfQuestion () – реалізую метод отримання списку питань з бази даних;
- getQuestionById () – реалізую метод отримання інформації конкретного питання;
- addNewQuestion () – реалізую метод який до питання з БД;
- editQuestion () – реалізую метод який редагує питання з БД.

1) Ім’я: userService
2) Відповідальність: реалізую методи отримання, зміну, відправку, видалення інформації про користувачів.
3) Атрибути: дані БД.
4) Операції:

- getListOfUsers () – реалізую метод отримання інформації про користувачів з БД;
- getUserByID () – реалізую метод отримання інформації про конкретного користувача з БД;
- getUserInfoByID () – реалізую метод отримання розгорнутої інформації про конкретного користувача з БД;
- postAddNewUser () – реалізую метод створення нового користувача у БД;
- postEditUsersById () – реалізую метод редагування інформації про конкретного користувача.

1) Ім’я: authController
2) Відповідальність: контролює процес авторизації, реєстрації, відновлення паролю користувача, перевірки даних вводу.
3) Атрибути: екземпляр класу, шаблон відповіді на запит, інформація про користувача, базу даних, шаблон помилок, валідатор.
4) Операції:

- register () – виконує запит авторизації;
- login () – виконує запит входу в систему;
- restore () – виконує запит зміни паролю;
- recoveryPass () – виконує запит відновлення паролю;
- changePass () виконує запит редагування паролю.

1) Ім’я: interviewController
2) Відповідальність: контролює процес взаємодії користувача з тестами.
3) Атрибути: шаблон відповіді на запит, шаблон помилок.
4) Операції:

- getListOfInterview () – виконує запит отримання інформації про тести;
- getInterviewById () – виконує запит отримання інформації про конкретний тест;
- addNewInterview () – виконує додавання нового теста;
- editInterview () – виконує запит на редагування теста;
- deleteInterview () виконує запит видалення теста.

1) Ім’я: interviewResultsController
2) Відповідальність: контролює процес взаємодії користувача з результатами тесту.
3) Атрибути: шаблон відповіді на запит, шаблон помилок, екземпляр класу.
4) Операції:

- getListOfInterviewResults () – виконує запит на отримання інформації про результатів тестів;
- getInterviewResultsById () – виконує запит на отримання інформації про результат тесту;
- postInterviewResults () – виконує додавання нового результату теста;

- postInterviewResultsById () – виконує запит на редагування результату тесту;

1) Ім’я: authService
2) Відповідальність: реалізую методи авторизації, входу у системи, зміни та відновлення паролю.
3) Атрибути: дані паролю, екземпляр класу.
4) Операції:

- dbConnect () – реалізую метод отримання інформації з БД;
- dbClose () – реалізую метод відключення БД;
- checkShowSql () – реалізую метод демонстрації SQL-запиту у БД;
- generateAccessToken () – реалізую метод створення токену;
- getUserByEmail () – реалізую метод пошуку користувача за поштою;
- login () – реалізую метод входу до системи;
- register () – реалізую метод реєстрації;
- changePass () – реалізую метод відновлення паролю.

1) Ім’я: interviewService
2) Відповідальність: реалізую методи взаємодії з тестами у БД
3) Атрибути: дані БД.
4) Операції:

- dbConnect () – реалізую метод отримання інформації з БД;
- getListOfInterview () – реалізую метод отримання списку тестів;
- getInterviewById () – реалізую метод отримання інформації про конкретний тест;
- addNewInterview () – реалізую метод додавання нового тесту;

- editInterview () – реалізую метод редакції тесту;
- deleteInterview () – реалізую метод видалення тесту.

1) Ім’я: interviewResultsService
2) Відповідальність: реалізую методи взаємодії з результатами тестів у БД
3) Атрибути: дані БД.
4) Операції:

- dbConnect () – реалізую метод отримання інформації з БД;
- getListOfInterviewResults () – реалізую метод отримання списку результатів тестів;
- getInterviewResultsById () – реалізую метод отримання результатів конкретного тесту;
- postInterviewResults () – реалізую метод додавання нового результату тестів;
- postInterviewResultsById () – реалізую метод додавання нового результату конкретного тесту.

2.2.3 Динамічна модель програмного забезпечення

Використовуючи побудовані статистичні моделі програмного забезпечення та сценарії використання була розроблена динамічна модель для веб-застосунку.

Діаграми послідовності є одним із способів формалізації сценаріїв використання. Її перевага заключається в тому, що на ранніх стадіях опису сценаріїв можливо з’ясувати склад взаємодіючих компонентів та описати потік повідомлень від одних компонентів до інших[2].

На рис. 2.14 представлено діаграму послідовності варіанту використання «Створення нового тесту».

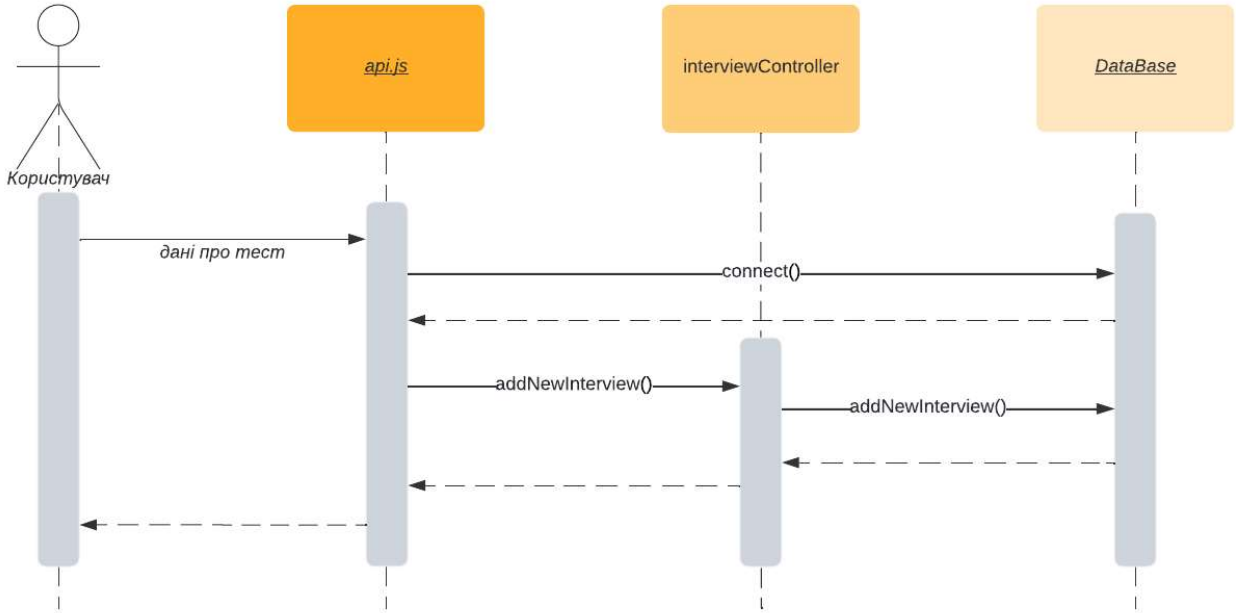

Рисунок 2.14 – Діаграма послідовності реалізації варіанту використання «Створення нового тесту»

Таким чином, на основі статистичні моделі програмного забезпечення та сценарії використання була розроблена діаграма послідовності реалізації варіанту використання «Створення нового тесту».

2.2.4 Логічна модель даних програмного забезпечення

Логічна модель – це модель даних предметної області, яка відображає реалізацію структури бази даних та не залежить від обраної СУБД[3]. Розроблена логічна модель програмного забезпечення для автоматизації процесу тестування та діагностики рівня професійних знань серед здобувачів вищої освіти на рис. 2.15.

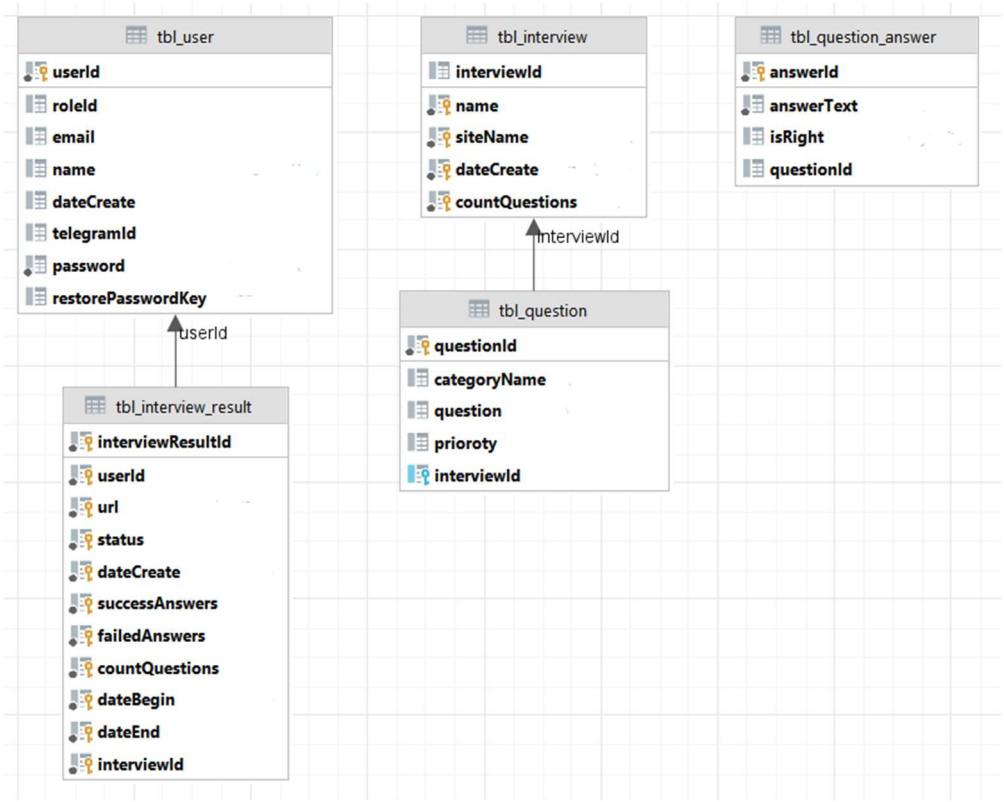

Рисунок 2.15 – Логічна модель структури бази даних

В результаті була розроблена логічна модель структури даних програмного забезпечення.

2.3 Робочий проект програмного забезпечення для автоматизації процесу тестування рівня професійних знань серед здобувачів вищої освіти

2.3.1 Аналіз та обґрунтування вибору мови програмування

Для розробки програмного забезпечення у вигляді веб-застосунку для автоматизації процесу тестування та діагностики рівня професійних знань серед здобувачів вищої освіти обрано мову програмування JavaScript та середовище розробки WebStorm[5].

JavaScript® (часто скорочують до JS) – це легка, інтерпретована, об'єктно-орієнтована мова з функціями першого класу, найвідоміша скриптова мова для веб-сторінок, але також використовується в багатьох не браузерних оточеннях. Прототипно-орієнтована, мультипарадигмальна

мова сценаріїв, яка підтримує динамічний, об'єктно-орієнтований, імперативний та функціональний стилі програмування[6]. Середовище програмування WebStorm — це інтегроване середовище розробки для кодування в JavaScript і пов’язаних з ним технологій, включаючи TypeScript, React, Vue, Angular, Node.js, HTML і таблиці стилів. Так само, як IntelliJ IDEA та інші IDE JetBrains, WebStorm робить ваш досвід розробки більш приємним, автоматизуючи рутинну роботу та допомагаючи з легкістю справлятися зі складними завданнями. WebStorm значну велику кількість можливостей для ефективної і швидкої роботи: розумне автодоповнення, аналіз коду в реальному часі і надійний рефакторинг. Отже, розробка веб-додатку для веб-браузерів мовою JS найбільш актуальна для даного проекту.

2.3.2 Побудова моделі реалізації

Модель реалізації складається з діаграми розгортання та діаграми компонентів. Діаграма реалізації відображає склад модулів системи та їх взаємодію. Діаграма розгортання відображає склад програмних об’єктів, компонентів, процесів. Діаграма компонентів відображає залежності складових частин програмного забезпечення.

2.3.2.1 Побудова діаграми компонентів

Діаграма компонентів відображає, як виглядає модель системи на фізичному рівні. На діаграмі зображені компоненти програмного забезпечення і їх взаємозв'язки. Діаграма компонентів зображена на рис. 2.16.

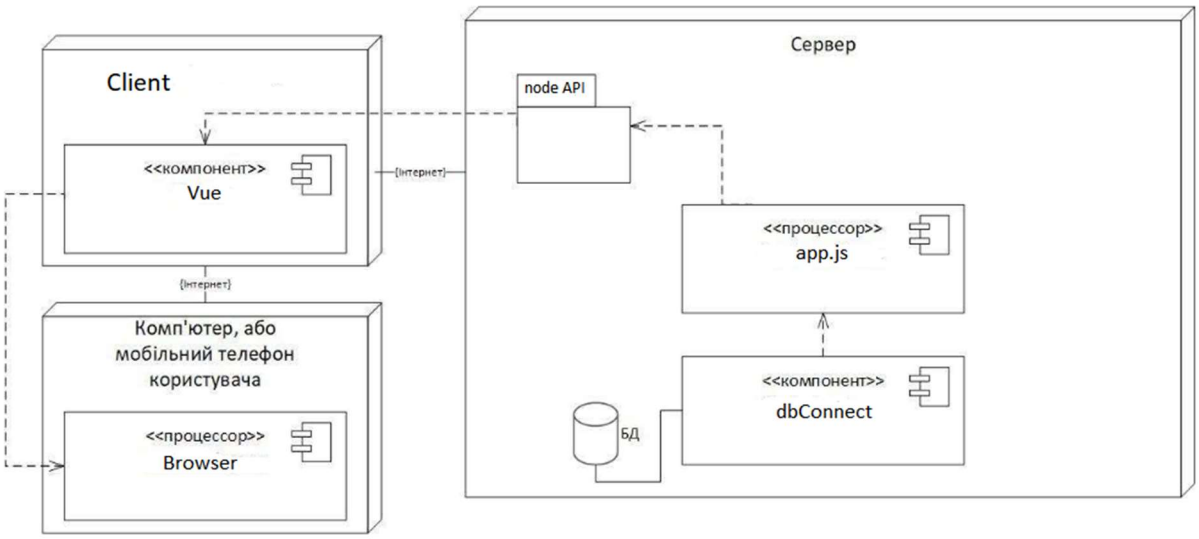

Рисунок 2.16 – Діаграма компонентів

У результаті побудовано діаграму компонентів, яка відображає залежності складових частин програмного забезпечення.

2.3.2.2 Побудова діаграми розгортання програми

Діаграма розгортання відображає фізичні взаємозв ' язки між і апаратними та програмними компонентами системи. Кожен вузол на діаграмі розміщення являє собою певний тип обчислювального пристрою, в більшості випадків – апаратні засоби. Діаграма розгортання зображена на рис. 2.17

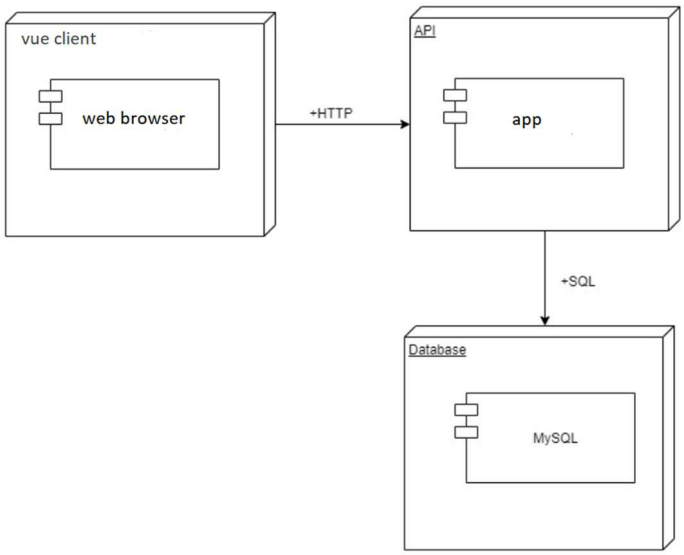

Рисунок 2.17 – Діаграма розгортання

Таким чином, було побудовано діаграму розгортання, що показує конфігурацію виконуваної програмної системи.

2.3.3 Процес тестування варіантів використання

У підрозділі проведено тестування основних варіантів використання «Авторизація», «Відновлення паролю», «Створення тесту».

Тестування варіанта використання «Авторизація»

Для тестування варіанту використання «Авторизація» був використаний метод еквівалентної розбивки. Виділені класи еквівалентності, представлені в табл. 2.1.

Таблиця 2.1 – Класи еквівалентності для тестування варіанту використання «Авторизація»

Вхідні дані	Правильні класи	Неправильні класи
Логін	Логін введено вірно (1)	Логін не було знайдено(2), Логін не введено (3)
Пароль	Пароль введено вірно (4)	Пароль не вірний(5), Пароль не було введено (6)

Результати тестування правильних класів еквівалентності представлені в табл. 2.2.

Таблиця 2.2 – Тести правильних класів еквівалентності для тестування варіанту
використання «Авторизація»

No тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
1	1, 4	Логін: Пароль:	Перехід на головну сторінку користувача	Тест пройдено

Тестування правильних класів еквівалентності для варіанту використання «Авторизація» успішно виконано.

Таблиця 2.3 – Тести не правильних класів еквівалентності для тестування варіанту використання «Авторизація»

No тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
1	2, 5	Логін: Пароль:	Повідомлення про помилку	Тест пройдено
2	3,6	Логін: Пароль:	Повідомлення про помилку	Тест пройдено

Тестування не правильних класів еквівалентності для варіанту використання «Авторизація» успішно виконано.

Таблиця 2.4 – Класи еквівалентності для тестування варіанту використання «Відновлення паролю»

Вхідні дані	Правильні класи	Неправильні класи
Пошта	Пошта введено вірно (1)	Пошту не було знайдено(2), Пошта не введено (3)

Результати тестування правильних класів еквівалентності представлені в табл. 2.5

Таблиця 2.5 – Тести правильних класів еквівалентності для тестування варіанту
використання «Відновлення паролю»

No тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
1	1	Пошта:	На указану пошту відправлено лист для відновлення паролю	Тест пройдено

Тестування правильних класів еквівалентності для варіанту використання «Авторизація» успішно виконано.

Таблиця 2.6 – Тести не правильних класів еквівалентності для тестування варіанту використання «Авторизація»

No тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
1	2	Пошта:	Повідомлення про помилку	Тест пройдено
2	3	Пошта:	Повідомлення про помилку	Тест пройдено

Тестування не правильних класів еквівалентності для варіанту використання «Відновлення паролю» успішно виконано.

Таблиця 2.7 – Класи еквівалентності для тестування варіанту використання «Створення нового тесту»

Вхідні дані	Правильні класи	Неправильні класи
Назва тесту	Назва тесту коректна (1)	Назва тесту не коректна (2), Назва тесту не введено (3)
Назва предмету	Предмет тестування обрано (4)	Предмет тестування не обрано(5)

Результати тестування правильних класів еквівалентності представлені в табл. 2.8

Таблиця 2.8 – Тести правильних класів еквівалентності для тестування варіанту використання «Створення нового тесту»

No тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
1	1, 4	Назва тесту: Назва предмету:	Тест створено і добавлено у БД	Тест пройдено

Тестування правильних класів еквівалентності для варіанту використання «Створення нового тесту» успішно виконано.

Таблиця 2.9 – Тести не правильних класів еквівалентності для тестування варіанту використання «Створення нового тесту»

No тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
1	2, 5	Назва тесту: Назва предмету:	Повідомлення про помилку	Тест пройдено
2	3	Назва тесту: Назва предмету:	Повідомлення про помилку	Тест пройдено

Тестування не правильних класів еквівалентності для варіанту використання «Створення нового тесту» успішно виконано.

2.3.4 Випробування програмного забезпечення

Випробування програми були виконані згідно з програмою та методикою випробувань, яка знаходиться у Додатку Д

1) Перевірка доступу до головної сторінки у якості Адміністратора

Після введення у відповідні поля даних авторизації адміністратора – відкривається головна сторінка веб-застосунку адміністратора, шо зображено на рис 2.18

Рисунок 2.18 – Головна сторінка адміністратора

2) Перевірка доступу до головної сторінки у якості Адміністратора

Після введення у відповідні поля даних авторизації студента – відкривається головна сторінка веб-застосунку студента, шо зображено на рис 2.19

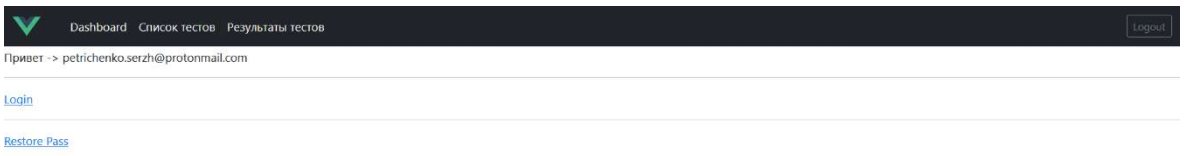

Рисунок 2.19 – Головна сторінка студента

3) Перевірка створення нового тесту

Після введення у відповідні поля даних про назву тесту, предмет тестування та натиснувши на кнопку “Додати тест” – тест зберігається та додається у базу даних, шо зображено на рис 2.20

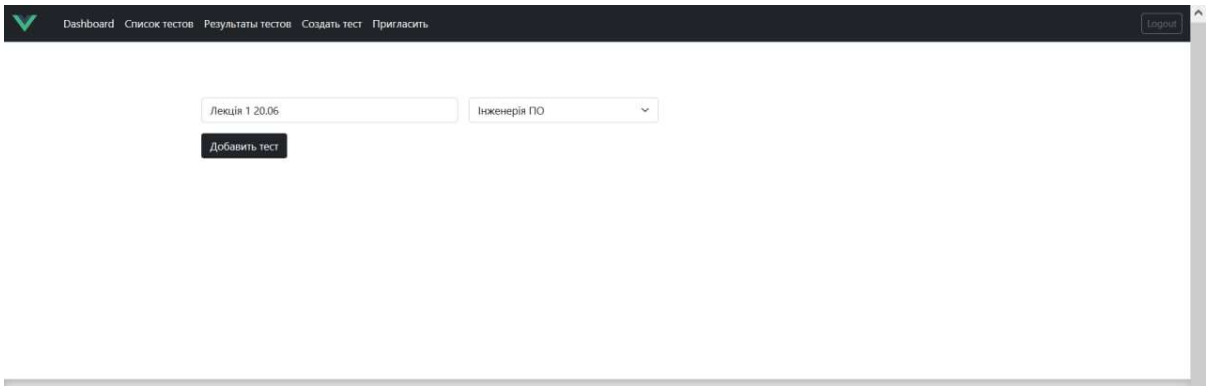

Рисунок 2.20 – Сторінка створення нового тесту

4) Перевірка проходження тесту

Після вибору відповідей на запитання тесту та натиснувши на кнопку “Виконано” – тест зберігається та його результати додається у базу даних, що зображено на рис 2.21

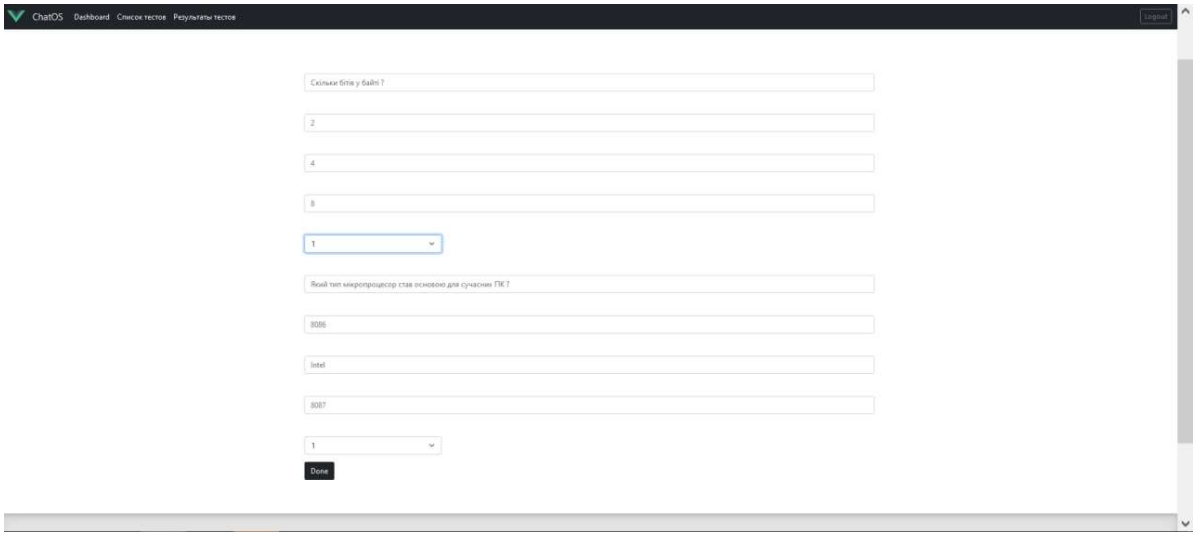

Рисунок 2.21 – Сторінка проходження тесту

3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В процесі виконання кваліфікаційної роботи було розроблено програмне забезпечення у вигляді веб-застосунку для автоматизації процесу тестування та діагностики рівня професійних знань серед здобувачів вищої освіти. Основними користувачами створення програмного забезпечення є адміністратор та студент.

Веб-застосунок розроблено мовою програмування JavaScript, оскільки JS є зручною та найчастіше використовується як мова сценаріїв для Веб.

Основні цілі розробки програмного забезпечення:

- Покращення взаємодії між студентом та викладачем в процесі оцінки рівня знань;
- полегшення зберігання та систематизація результатів тестів;
- створення легкодоступної веб-системи оцінки рівня знань для віддаленого використання студентами.

У процесі виконання кваліфікаційної роботи були створені наступні моделі:

- статична модель;
- інформаційна модель;
- модель варіантів використання;
- модель реалізації;
- динамічна модель.

Веб-застосунок націлений на використання широкою аудиторією

студентів, що не є професійними програмістами. Оскільки для запуску додатку необхідно лише доступ до мережі Інтернет та сучасний браузер, його можливо та зручно використовувати як за допомогою доступних ПК, так і за допомогою смартфонів.

Тестування веб-застосунку для тестування та діагностики рівня професійних знань здобувачів вищої освіти показало, що програмне забезпечення успішно виконує усі поставленні завдання.

У кваліфікаційній роботі представлені положення та норми охорони праці, а також був проведений аналіз небезпечних і шкідливих факторів та розробка заходів щодо забезпечення сприятливих умов праці.

Текст програми наведено у додатку Б. Опис програми представлено в додатку В. Інструкція користувача наведена у додатку Г. Методика випробувань представлена у додатку Д.

В ході роботи проаналізовано аналогічні варіанти програмного забезпечення, що існують у вільному доступі, результати аналізу було взято до уваги при розробці веб-застосунку

Результати кваліфікаційної роботи відповідають поставленій меті.

4 ОХОРОНА ПРАЦІ

4.1 Вступна частина

Охорона праці – система забезпечення безпеки життя і здоров'я робітників в процесі трудової діяльності, включаючи правові, соціально-економічні, санітарно-
гігієнічні та лікувально-профілактичні, реабілітаційні та інші заходи. Функціями охорони праці є дослідження санітарії та гігієни праці, проведення заходів щодо зниження впливу шкідливих факторів на організм працівників в процесі праці.

Основним методом охорони праці є використання техніки безпеки. При цьому вирішуються два основних завдання: створення машин та інструментів, при роботі з якими виключена безпека для людини, та розробка спеціальних засобів захисту,що забезпечують безпеку людини в процесі праці, також проводиться навчання працюючих безпечним прийомам праці та використання засобів захисту, створюються умови для безпечної роботи.

Основна мета покращення умов праці – досягнення соціального ефекту, тобто забезпечення безпеки праці, збереження життя та здоров'я працюючих, скорочення кількості нещасних випадків та захворювань на виробництві. Покращення умов праці дає і економічні результати: ріст прибутку (у зв'язку з підвищенням продуктивності праці); скорочення витрат, пов'язаних з компенсацією за роботу з шкідливими та тяжкими умовами праці, зменшення втрат пов'язаних з травматизмом, професійною захворюваністю.

Робоче місце – просторова зона, оснащена необхідними засобами, в якій здійснюється трудова діяльність робітника або групи робітників, які спільно виконують виробничі завдання. Робоче місце є частиною виробничо-технологічної структури підприємства (організації), воно призначене для виконання частини

технологічного (виробничого) процесу та визначається на основі трудових та інших діючих норм та нормативів.

Робоча зона – простір, обмежений по висоті двома метрами над рівнем підлоги та площадки, яких знаходяться місця постійного або непостійного (тимчасового) перебування працюючих. До постійних належать місця, на яких працюючий знаходиться понад 50% робочого часу протягом зміни або більше двох годин безперервно. Якщо робота здійснюється в різних пунктах робочої зони, то постійним робочим місцем вважається вся робоча зона. Умови праці – сукупність факторів виробничого середовища, що впливають на здоров'я та працездатність людини в процесі праці. Дослідження умов праці показали, що факторами виробничого середовища в процесі праці є:

- санітарно-гігієнічна обстановка, що визначає зовнішнє середовище в робочій зоні – мікроклімат, механічне коливання, випромінювання, температуру, освітлення та інші;
- психофізіологічні елементи: робоча поза, фізичне навантаження, нервово-психологічна напруга та інші, які обумовлені процесом праці;
- соціально-психологічні елементи, що складають характеристику так званого психологічного клімату.

Професійним захворюванням називають захворювання, викликане впливомшкідливих умов праці. До них належать: хронічні пилові бронхіти, вібраційнахвороба, отруєння різними токсичними речовинами та інші професійнізахворювання в залежності від тяжкості та строків виявлення, можуть супроводжуватися та не супроводжуватися втратою працездатності. У важких випадках вони можуть призвести до інвалідності.

До основних функцій управління охороною праці належать:

- облік показників стану умов і безпеки праці;
- аналіз та оцінка стану умов і безпеки праці;
- прогнозування і планування робіт, їх фінансування;

- організація та координація робіт;

4.2 Вимоги з охорони праці при роботі з обчислювальною технікою

Наймач, який використовує працю свого колективу, зобов'язаний зробити їх робочі місця максимально безпечними та комфортними. Розмір одного робочого місця має бути не менше 6 м ². При необхідності, суміжні робочі місця працівників, які виконують завдання на ПК, слід розділити перегородками заввишки до 2 метрів. При визначенні достатнього розміру приміщення на одну особу необхідно додатково враховувати шафи, тумби або інші предмети меблів або обладнання, що знаходяться у кімнаті. На кожному столі можна розташувати допоміжну оргтехніку (колонки, принтер, сканер), полиці для зберігання документів. Така кількість додаткових елементів не повинна перешкоджати видимості екрана. У разі надмірного шуму або вібрації технічного обладнання, сторона, що наймає, зобов'язана забезпечити трудовий колектив антивібраційними килимками. Робочий стілець співробітника повинен бути підйомно-поворотним, легко регульованим по висоті та забезпечувати належну підтримку та зручне положення спини та хребта людини. Щодня необхідно проводити вологе прибирання приміщення, та очищати робоче місце та безпосередньо монітор ПК від запиленості.

При прийнятті на роботу кожна особа має пройти лікарський огляд. Крім того, при подальшій трудовій діяльності в компанії, така особа підлягає регулярному лікарському огляду не рідше ніж один раз на 2 роки. Обов'язковим є проходження таких лікарів як терапевта, невропатолога та офтальмолога. У компанії мають бути чітко встановлені перерви для відпочинку працівників (окрім обідньої), як правило, тривалістю 10-15 хвилин раз на годину або дві, в залежності від складності роботи. У будь-якому випадку, роботодавець повинен передбачити такий розпорядок роботи на підприємстві, щоб година безперервної роботи з комп'ютером

була не більше ніж 4 години. Додатково, для збереження належного рівня здоров'я та професійної придатності робітників, рекомендується виділити на підприємстві окреме побутове приміщення для відпочинку працівників та зняття ними нервово-емоційного напруження, що виникає при роботі з комп'ютером.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було проаналізовано процес проходження тестів та отримання кінцевих результатів, сформульовано задачі, проаналізовано наявні рішення на доступному ринку програмного забезпечення, проаналізовано доцільність розробки власного програмного забезпечення у вигляді веб-застосунку, побудовано інформаційну модель, модель варіантів використання та реалізувано ПЗ згідно заявленої теми.

У результаті аналізу існуючих програмних засобів у вигляді веб-застосунків «Google Forms» та «Typeform» було виявлено, що вони не відповідають усім наявним вимогам навчального закладу, а саме не мають специфічної направленості. Результатом аналізу – є рішення про початок розробки програмного забезпечення, що буде відповідати усім вимогам та потребам. У рамках кваліфікаційної роботи було проаналізовано предметну галузь, а також виконано розділ з охорони праці.

Аналізуючи методології тестування та існуючих програм було зроблено висновок про доцільність розробки нового програмного продукту, а також сформовано постановку задачі та виділено основні функції програмного забезпечення для автоматизації тестування.

У розділі з охорони праці було розглянуто основні положення про охорону праці та проаналізовано вимоги з охорони праці при роботі з обчислювальною технікою.

Програмне забезпечення кваліфікаційної роботи за темою «Розробка веб-застосунку для тестування та діагностики рівня професійних знань здобувачів вищої освіти» реалізоване згідно технічному завданню.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Електронні засоби навчання *[Електронний ресурс] - Режим до*ступу:https://www.gstu.by/sites/default/files/atoms/files/d8/73/zayac-tezisy1.pdf.

2. Застосування UML. Діаграма послідовності [Електронний ресурс] -Режим доступу: https://dut.edu.ua/ua/news-1-626-7897-zastosuvannya-uml-chastina-2-diagrama-poslidovnosti---sequence-diagram_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy

3. Логічна модель *[Електронний ресурс]* - https://stud.com.ua/121128/informatika/sistemno_dinamichni_model i_organizatsiy

4. Опис веб-застосунку «Google Forms» [Електронний ресурс] - Режим доступу: https://workspace.google.com/products/forms/?hl=en.

5. Опис веб-застосунку «WebStorm» [Електронний ресурс] - Режим доступу:https://www.jetbrains.com/help/webstorm/getting-started-with webstorm.html

6. Опис мови програмування «JavaScript» [Електронний ресурс] - Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript

7. Технології створення якісних і ефективних тестів *[Електронний ресурс] - Режим до*ступу: https://www.znanius.com/5520.html.

ДОДАТОК А - ТЕХНІЧНЕ ЗАВДАННЯ

1. Вступ

Назва програмного забезпечення: "Веб-застосунку для тестування та діагностики рівня професійних знань здобувачів вищої освіти".

Призначення програми: Проведення педагогічних тестів серед студентів з метою визначення рівня професійних знань.

2.Підстави для розробки

Підставами для розробки даного програмного забезпечення є відсутність аналогів, які би в повній мірі реалізовували всі необхідні функції для успішної взаємодії студента та викладача, зручного та швидкого проходження оцінки професійних знань.

3.Призначення розробки.

3.1. Функціональне призначення.

Функціональним призначенням програми є автоматизація процесу діагностики рівня професійних знань серед студентів.

3.2. Експлуатаційне призначення.

Програма повинна експлуатуватися в браузері такими людьми як:
- викладач(адміністратор). Запрошення нових студентів, додавання, редагування тестів, перевірка результатів;
- студент. Проходження тестів, перевірка результатів тестувань;

4. Вимоги до програмного забезпечення

4.1. Вимоги до інструментів розробки

Коди програми повинні бути реалізовані на мові JavaScript. В якості середовища розробки повинен бути використан Webstorm.
В якості бази даних використовувати СУБД MySQL версії 8.0.0.

4.2. Вимоги до виконуваних функцій

Для адміністратора мають бути реалізовані наступні функції:

- Створення нових тестів;
- редагування тестів;
- видалення тестів;
- створення нових запитань;
- редагування запитань;
- видалення запитань;
- створення відповідей на запитання;
- видалення відповідей на запитання;
- редагування відповідей на запитання;
- перегляд користувачів;
- перегляд результатів тестування користувачів.

Для користувача мають бути реалізовані наступні функції:

- Перегляд доступних тестів;
- проходження доступних тестів;
- перегляд результатів тестів;
- перегляд загальної статистики тестування.

4.3. Вимоги до організації вхідних та вихідних даних

Введення інформації здійснюється за допомогою миші, клвіатури або сенсорного єкрану. Вихідна інформація відображається за єкрану ПК або смартфону.

4.4. Вимоги до надійності

4.4.1. Вимоги до забезпечення надійного функціонування програми

Надійне (стійке) функціонування програми має бути забезпечене виконанням сукупності організаційно-технічних заходів, перелік яких наведено нижче:

- організацією безперебійного живлення технічних засобів;
- використанням ліцензійного програмного забезпечення.

4.4.2. Час відновлення після відмови.

Час відновлення після відмови, викликаної збоєм електроживлення на сервері (іншими зовнішніми факторами), не фатальним збоєм операційної системи, не повинно перевищувати 5-ти хвилин.

Час відновлення після збою, викликаного несправністю технічних засобів, фатальним збом операційної системи, не повинно перевищувати час, необхідний для усунення несправностей технічних засобів.

4.4.3. Відмови через некоректність дій користувача.

Відмови програми можливі внаслідок некоректних дій користувача при відправленні даних. При введені некоректних даних програма не буде виконувати запит користувача.

4.5. Вимоги до видів обслуговування.

Програма не потребує проведення спеціалізованих видів обслуговування.

4.6. Вимоги до чисельності та кваліфікації персоналу. Мінімальна кількість персоналу, необхідна для функціонування програми, повинна складати одну особу — адміністратор(викладач), що має мати базові навички роботи з комп’ютером.

4.7. Вимоги до складу і параметрів технічних засобів.

4.7.1. Backend.

В склад технічних засобів повинен входити сервер з наступними параметрами:

- операційна система: Linux;
- процесор: Intel Celeron G, 3.1 GHZ;
- інтернет: 10 Мбит/с та більше;
- оперативна пам’ять: мінімум 2Гб;
- твердий диск: для нормальної роботи системі потрібно 50Мб вільного дискового простору.

4.7.2. Frontend.

В склад технічних засобів повинен входити комп’ютер або смартфон з мінімальними характеристиками, який підтримує встановлення сучасного браузера з наступними параметрами:

- операційна система: Android 5 +, Windows XP, 7, Vista, 8, 8.1, 10;

- процесор: Intel Pentium 4/Athlon 64 або пізнішої версії з підтримкою SSE2;
- інтернет: 10 Мбит/с та більше;
- оперативна пам’ять: мінімум 512Гб;
- твердий диск: 350 Мб+.

4.8. Вимоги до інформаційної і програмної сумісності.

4.8.1. Вимоги до інформаційних структур і методів розв’язку.

Вимоги до інформаційних структур (файлів) на вході та виході, а також до методів вирішення не висуваються.

5. Вимоги до програмної документації.

Попередній склад програмної документації повинен включати в себе:

- технічне завдання;
- текс програмних модулів;
- програма та методика випробовування;
- опис програми;
- інструкція користувача.

6. Техніко-економічні показники.

Техніко-економічні показники не розраховуються.

7.Стадії та етапи розробки.

Стадії та етапи розробки представлено в таблиці 1.

Таблиця 1 – Стадії та етапи розробки

No	Стадії розробки	Етапи розробки	Зміст робіт по етапах	Строки виконання
1	2	3	4	5
1	Технічне завдання	розробка, узгодження та затвердження технічного завдання	Постановка задачі; визначення та уточнення вимог до технічних засобів; визначення вимог до програми; визначення стадій, етапів і термінів розробки програми і документації на неї; узгодження і затвердження технічного завдання.	Початок 31.03.22 Закінчення 27.04.22
2	Ескізний проект	Розробка та затвердження ескізного проекту	Виявлення вимог, концепція майбутньої системи, складання ескізної частини програмного продукту	Початок 27.04.22 Закінчення 06. 05.22
3	Технічний проект	Розробка та затвердження технічного проекту	Загальні системні вирішення, алгоритми задач, оцінка економічної ефективності автоматизованої системи та перелік заходів підготовки об’єкта для провадження	Початок 06.05.22 Закінчення 13.05.22
4	Робочий проект	Розробка та затвердження робочого проекту	Деталізовані загальносистемні проекті рішення, програми та інструкції для розв’язку завдань, розробка програмного забезпечення.	Початок: 13.05.22 Закінчення 09.06.22

ДОДАТОК Б - ТЕКСТ ПРОГРАМИ

Лістинг основних класів програми наведено нижче.

Лістинг класу app

```
const process = require('process');
process.stdin.resume();
process.on('SIGINT', () => {
    process.exit(1);
});

const env = process.env.NODE_ENV || 'development';
require('dotenv').config();

const express = require('express');
const exphbs = require('express-handlebars');
const url = require('url');
const jwt = require('jsonwebtoken');

const interviewRoutes = require('./routes/interview');
const interviewResultsRoutes =
require('./routes/interviewResults');
const questionRoutes = require('./routes/question');
const questionAnswerRoutes =
require('./routes/questionAnswer');
const authRoutes = require('./routes/auth');
const userRoutes = require('./routes/user');
const app = express();
var handlebars = exphbs.create({
    defaultLayout: 'main',
```

```
    extname: 'hbs'
});
app.engine('hbs', handlebars.engine);

app.enable('view cache');

app.set('views', './views');
app.set('view engine', 'hbs');

app.use(express.static(`${__dirname}/assets`));
app.use(express.urlencoded({ extended: true }));
app.use(express.json

app.use((req, res, next) => {
    res.setHeader('Access-Control-Allow-Origin', '*');
    console.log('------------------------request',
req.originalUrl);
    if (req.method === 'OPTIONS') {
        return res.status(200).setHeader('Access-
Control-Allow-Methods', 'POST').setHeader('Access-
Control-Allow-Headers', req.headers['access-control-
request-headers']).end();
    }
    const queryObject = url.parse(req.url, true).query;
    if (process.env.ENVIRONMENT != 'production') {
        process.env.SHOW_SQL = '';
        process.env.IS_ASK_SQL = 0;
        if (typeof queryObject.showSql !== 'undefined')
{
            process.env.IS_ASK_SQL = 1;
```

```
        }
    }
    if (typeof req.headers.authorization !==
'undefined') {
        const token = req.headers.authorization.split('
')[1];
        if (token) {
            var decodedData = false;
            try {
                decodedData = jwt.verify(token,
process.env.TOKEN_KEY);
            } catch (e) {}
            if (decodedData) {
                req.userInfo = decodedData;
            }
        }
    }
    next();
});
app.get('/', (req, res) => {
    res.render('home', {
        title: 'Greetings form White Rabbit',
        layout: false
    });
});

app.use('/api/interview', interviewRoutes);
app.use('/api/interviewResults',
interviewResultsRoutes);
app.use('/api/question', questionRoutes);
```

```
app.use('/api/questionAnswer', questionAnswerRoutes);
app.use(authRoutes);
app.use('/api/user', userRoutes);

app.use((req, res, next) => {
    console.log('res', res);
    //obj.sql = JSON.parse(process.env.SHOW_SQL);
    next();
});

async function start() {
    try {
        app.listen(process.env.PORT, process.env.HOST,
function () {
            console.log(`Server listens
http://${process.env.HOST}:${process.env.PORT}`);
        });
    } catch (e) {
        console.error(e);
    }
}

start();
```

Лістинг класу dbConnect

```
const db = require('mysql2/promise');
const errorHandler = require("../utils/ErrorHandler");
class dbConnect {
    conn = false;
```

```
    table_prefix = 'tbl_';

    async connect() {
        try {
            this.conn = await db.createConnection({
                host: process.env.DB_HOST,
                port: process.env.DB_PORT,
                user: process.env.DB_USER,
                password: process.env.DB_PASS,
                database: process.env.DB_NAME
            });
        }
        catch (e) {
        }
    }
    // connect(){
    //     this.conn.connect();
    //     return this.conn;
    // }
}
module.exports = dbConnect;
```

Лістинг класу authController

```
var authService = require('../services/authService.js');
var generalResponse =
require('../routes/generalResponse.js');
const nodemailer = require('nodemailer');
const { validationResult } = require("express-
validator");
```

```
const errorHandler = require('../utils/ErrorHandler')
let transporter = nodemailer.createTransport({
    service: 'gmail',
    auth: {
        user: 'sedutest123@gmail1.com',
        pass: '151920224455'
    },
});
class authController {
    async register(req, res) {
        const validationErrors = validationResult(req);
        if (!validationErrors.isEmpty()) {
            return res.status(400).json({ message:
"Некорректные данные", validationErrors });
        }
        try {
            var {email} = req.body;
            var pass = req.body.pass || '';
            var obj = generalResponse.responseBody();
            var check = await
authService.register(email, pass);
            if (check === true) {
                obj.error = false;
                obj.message = 'done';
            } else {
                obj.message = "Error: " + check;
            }
            if (process.env.SHOW_SQL) {
                obj.sql =
JSON.parse(process.env.SHOW_SQL);
```

```
            }

res.status(generalResponse.responseStatus.success).json(
obj);
        } catch (e) {
            errorHandler(res,e);
        }

    }

    async login(req, res) {
        const validationErrors = validationResult(req);
        if (!validationErrors.isEmpty()) {
            return res.status(400).json({ message:
"Некорректные данные", validationErrors });
        }
        try {
            var {email, pass} = req.body;

            var obj = generalResponse.responseBody();
            obj.error = false;
            obj.message = 'done';
            var check = await authService.login(email,
pass);
            if (check.error) {
                return
res.status(generalResponse.responseStatus.notAuth).json(
{message: "Error: " + check.message});
            } else {
                obj.data = {};
```

```
                obj.data.token = check.token;
            }
            if (process.env.SHOW_SQL) {
                obj.sql =
JSON.parse(process.env.SHOW_SQL);
            }
            res.setHeader('Access-Control-Allow-
Headers', 'origin, content-type, accept');

res.status(generalResponse.responseStatus.success).heade
r({"Authorization": "Bearer " +
obj.data.token}).json(obj);
        } catch (e) {
            errorHandler(res,e);
        }
    }

    async restore(req, res) {
        const validationErrors = validationResult(req);
        if (!validationErrors.isEmpty()) {
            return res.status(400).json({ message:
"Некорректные данные", validationErrors });
        }
        try {
            var { email } = req.body;
            var obj = generalResponse.responseBody();
            obj.error = false;
            obj.message = 'done';
            var check = await
authService.generateRecoverKey(email);
```

```
            if (check.error) {
                return res.status(400).json({ message:
check.message });
            } else {
                obj.data = {};
                obj.data.recoveryToken =
check.recoveryToken;
                obj.data.message = check.message;
                var info = await transporter.sendMail({
                    from: 'sedutest1213@gmail.com',
                    to: email,
                    subject: "Reovery link",
                    text: "Recovery link:" +
check.message,
                    html: ` <a
href="${check.message}">put</a>`,
                });
                console.log("Message sent: %s",
info.messageId);
                console.log("Preview URL: %s",
nodemailer.getTestMessageUrl(info));
                obj.data.link =
nodemailer.getTestMessageUrl(info);
            }
            if (process.env.SHOW_SQL) {
                obj.sql =
JSON.parse(process.env.SHOW_SQL);
            }
```

```
res.status(generalResponse.responseStatus.success).heade
r({
                "Recovery-Token": "Bearer " +
obj.data.recoveryToken,
                "Message": obj.data.message,
                "Link":
nodemailer.getTestMessageUrl(info)
            }).json(obj);
        } catch (e) {
            errorHandler(res,e);
        }
    }
    async recoveryPass(req, res) {
        try {
            var obj = generalResponse.responseBody();
            obj.error = false;
            obj.message = 'done';
            obj.data = await
authService.recoveryPass(req.params.recoveryToken);
            if (process.env.SHOW_SQL) {
                obj.sql =
JSON.parse(process.env.SHOW_SQL);
            }

res.status(generalResponse.responseStatus.success).json(
obj);
        } catch (e) {
            console.log(e);
        }
```

```
res.status(generalResponse.responseStatus.success).heade
r({"Authorization": "Bearer " + obj.data.token,
        'Access-Control-Allow-Headers': 'origin,
content-type, accept',
        'Access-Control-Allow-Origin': '*'}).json(obj);
    }
    async changePass(req, res) {
        const validationErrors = validationResult(req);
        if (!validationErrors.isEmpty()) {
            return res.status(400).json({ message:
"Некорректные данные", validationErrors });
        }
        try {
            var { newPass, token } = req.body;
            var obj = generalResponse.responseBody();
            obj.error = false;
            obj.message = 'done';
            obj.data = await
authService.changePass(newPass, recoveryToken);
            if (process.env.SHOW_SQL) {
                obj.sql =
JSON.parse(process.env.SHOW_SQL);
            }


res.status(generalResponse.responseStatus.success).json(
obj);
        } catch (error) {
            return error
```

```
        }
        await sendAccess(newPass, obj.data.email)
    }

}
async function sendAccess(pass, email) {
    var info = await transporter.sendMail({
        from: 'sedutest123@gmail.com',
        to: email,
        subject: "Acess data",
        text: "Email: " + email,
        html: `Email: ` + email +`<br/>` + ` Password: `
+ pass,
    });
    console.log("Message sent: %s", info.messageId);
    console.log("Preview URL: %s",
nodemailer.getTestMessageUrl(info));
}

module.exports = new authController();
```

Лістинг класу interviewController

```
var interviewService =
require('../services/interviewService.js');
var generalResponse =
require('../routes/generalResponse.js');
const errorHandler = require("../utils/ErrorHandler");
class interviewController {
    async getListOfInterview(req, res) {
```

```
        var obj = generalResponse.responseBody();
        if (typeof req.userInfo == 'undefined') {
            obj.message = тще;
            return
res.status(generalResponse.responseStatus.notAuth).json(
obj);
        }
        obj.error = false;
        obj.message = 'done';
        try{
            obj.data = await
interviewService.getListOfInterview();
            if (process.env.SHOW_SQL) {
                obj.sql =
JSON.parse(process.env.SHOW_SQL);
            }

res.status(generalResponse.responseStatus.success).json(
obj);
        }
        catch (e) {
            errorHandler(res,e);
        }
    }

    async getInterviewById(req, res) {
        var obj = generalResponse.responseBody();
        obj.error = false;
        obj.message = 'done';
        try{
```

```
            obj.data = await
interviewService.getInterviewById(req.params.id);
            if (process.env.SHOW_SQL) {
                obj.sql =
JSON.parse(process.env.SHOW_SQL);
            }

res.status(generalResponse.responseStatus.success).json(
obj);
        }
        catch (e) {
            errorHandler(res,e);
        }
    }
    async addNewInterview(req, res) {
        var { name, siteName } = req.body;
        var obj = generalResponse.responseBody();
        obj.error = false;
        obj.message = 'done';
        try {
            obj.data = await
interviewService.addNewInterview(name, siteName);

            if (process.env.SHOW_SQL){
                obj.sql =
JSON.parse(process.env.SHOW_SQL);
            }

res.status(generalResponse.responseStatus.success).json(
obj);
```

```
        }
        catch (e) {
            errorHandler(res,e);
        }
    }
    async editInterview(req, res) {
        var { id, name, siteName } = req.body;
        var obj = generalResponse.responseBody();
        obj.error = false;
        obj.message = 'done';
        await interviewService.editInterview(id, name,
siteName);
        obj.data = req.body;

res.status(generalResponse.responseStatus.success).json(
obj);
    }
    async deleteInterview(req, res) {
        var { id, name, siteName } = req.body;
        var obj = generalResponse.responseBody();
        obj.error = false;
        obj.message = 'done';
        await interviewService.deleteInterview(id, name,
siteName);
        obj.data = req.body;

res.status(generalResponse.responseStatus.success).json(
obj);
    }
}
```

```
module.exports = new interviewController();
```

Лістинг класу authService

```
var dbClass = require('../controllers/dbConnect.js');
const db = require('mysql2/promise');
const bcrypt = require("bcryptjs");
const passGenerator = require('generate-password');
const jwt = require("jsonwebtoken");
class authService {
    async dbConnect() {
        var conn = await db.createConnection({
            host: process.env.DB_HOST,
            port: process.env.DB_PORT,
            user: process.env.DB_USER,
            password: process.env.DB_PASS,
            database: process.env.DB_NAME
        });
        return conn;
    }
    dbClose(conn) {
        conn.end((err) => {
            if (err) {
                console.error(err);
                return err;
            }
        });
    }
    checkShowSql(title, sql) {
        if (Boolean(Number(process.env.IS_ASK_SQL))) {
```

```
            var sqlQuery = { title: title, sql: sql };
            var json = process.env.SHOW_SQL ?
JSON.parse(process.env.SHOW_SQL) : [];
            json.push(sqlQuery);
            process.env.SHOW_SQL =
JSON.stringify(json);}}
    generateAccessToken(id, roleId) {
        const payload = {
            id, roleId
        };
        return jwt.sign(payload, process.env.TOKEN_KEY,
{ expiresIn: "24h" });
    }

    generateRecoveryToken(email) {
        const payload = {
            email
        };
        return jwt.sign(payload, process.env.TOKEN_KEY,
{ expiresIn: "24h" });
    }

    async login(email, pass) {
        var conn = await this.dbConnect();
        var user = await this.getUserByEmail(email);
        if (!user) {
            return { error: true, message: "Неправильный
логин или пароль" };
        }
```

```
        const checkPass = bcrypt.compareSync(pass,
user.password);
        if (checkPass) {
            const token =
this.generateAccessToken(user.userId, user.roleId);
            return { error: false, token: token };
        } else {
            return { error: true, message: "Error" };
        }
    }

    async getUserByEmail(email, pass = '') {
        var conn = await this.dbConnect();
        var sql = `SELECT * FROM tbl_user WHERE email =
'${email}'`;
        if (pass) {
            const hashPass = bcrypt.hashSync(pass, 7);
            sql += ` AND password = '${hashPass}'`;
        }
        this.checkShowSql(sql);
        const [rows, fields] = await conn.execute(sql);
        this.dbClose(conn);
        return rows[0] || false;
    }

    async register(email, pass = '') {
        var conn = await this.dbConnect();
        pass = pass || passGenerator.generate({ length:
10, numbers: true });;
        console.log('pass', pass);
```

```
        const hashPass = bcrypt.hashSync(pass, 7);
        if (!(await this.getUserByEmail(email))) {
            var sql = `INSERT INTO tbl_user (roleId,
email, password)
                VALUES (1, '${email}', '${hashPass}')`;
            this.checkShowSql( sql);
            const [rows, fields] = await
conn.execute(sql);
            this.dbClose(conn);
            return true;
        } else {
            return email + done';
        }
    }
    async generateRecoverKey(email) {
        var conn = await this.dbConnect();
        var user = await this.getUserByEmail(email);
        const recoveryToken =
this.generateRecoveryToken(email);
        if (user) {
            return {
                error: false,
                message: 'http://' + process.env.HOST +
":" + process.env.PORT + '/' + 'recoveryPass/' +
recoveryToken,
                recoveryToken: recoveryToken
            }
        } else {
            return {
                error: true,
```

```
                message: Email: '  email
            }
        }
    }
    recoveryPass(recoveryToken) {
        try {
            var decoded = jwt.verify(recoveryToken,
process.env.RECOVERY_TOKEN_KEY);
            return {
                recoveryToken: recoveryToken,
                error: false,
                message: decoded
            }
        } catch (error) {
            return error
        }
    }
    async changePass(newPass, recoveryToken) {
        try {
            var conn = await this.dbConnect();
            var decoded = jwt.verify(recoveryToken,
process.env.RECOVERY_TOKEN_KEY);
            const hashPass = bcrypt.hashSync(newPass,
7);
            var sql = `UPDATE tbl_user SET password =
'${hashPass}' WHERE email = '${decoded.email}'`;
            this.checkShowSql(sql);
            const rows = await conn.execute(sql);
            this.dbClose(conn);
            return {
```

```
                success: true,
                password: newPass,
                email: decoded.email
            }
        } catch (error) {
            return {
                success: false
            }
        }
    }
}
module.exports = new authService();
```

ДОДАТОК В - ОПИС ПРОГРАМИ

1 Загальні відомості

1.1 Позначення і найменування програми

Найменування: Програмне забезпечення для автоматизації процесу тестування та діагностики рівня професійних знань серед здобувачів вищої освіти

1.2 Програмне забезпечення, необхідне для функціонування програми

Щоб запустити веб-застосунок для автоматизації процесу тестування та діагностики рівня професійних знань серед здобувачів вищої освіти необхідно мати встановлений браузер загального використання на планшеті, смартфоні або персональному ПК з будь-якою операційною системою.

1.3 Мова програмування

Програмне забезпечення у вигляді веб-застосунку для автоматизації процесу тестування та діагностики рівня професійних знань серед здобувачів вищої освіти розроблено мовою програмування JavaScript.

2.1 Класи розв'язуваних завдань та призначення програми

Програмне забезпечення у вигляді веб-застосунку для автоматизації процесу тестування та діагностики рівня професійних знань серед здобувачів вищої

освіти призначене для полегшення та спрощення проведення оцінювання рівня знань серед студентів. Призначенням є
автоматизація тестування, збір та аналіз результатів оцінювань.
Програмне забезпечення на етапі авторизації визначає роль користувача згідно його логіну у системі. В системі передбачені такі ролі: студент, адміністратор.
Для студент мають бути реалізовані наступні функції:

- Проходження тестів;
- вибір тесту;
- перегляд минулих результатів тестування.

Для адміністратора мають бути реалізовані наступні функції:

- Запрошення нового користувача;
- перегляд списку користувачів;
- перегляд списку тестів;
- редагування, видалення тестів;
- перегляд результатів тестів.

2.2 Функціональні обмеження на застосування

Для роботи з веб-застосунком необхідне підключення до мережі Інтернет та ПК або смартфон який підтримує сучасні браузери.

3 Опис логічної структури

3.1 Алгоритм програми

Програмне забезпечення для автоматизації процесу тестування та діагностики рівня професійних знань серед здобувачів вищої освіти у вигляді веб-застосунку. Веб-застосунок виступає зовнішнім клієнтом та через веб сервер взаємодіє з базою даних.

Алгоритм роботи програми:

- Користувач використовуючи браузер переходить на сторінку застосунку;
- веб-застосунок відсилає запит на сервер;
- сервер обробляє поступаючі дані та повертає результат;
- результат відповіді відображається на vue-клієнту.

3.2 Використовувані методи

Програмне забезпечення у вигляді веб-застосунку для автоматизації процесу тестування та діагностики рівня професійних знань серед здобувачів вищої освіти розроблено мовою сценарієв програмування JavaScript.
В основу розробки було покладено об’єктно-орієнтовану модель.

3.3 Структура програми

Програмне забезпечення для автоматизації процесу тестування та діагностики рівня професійних знань серед здобувачів вищої освіти є веб-

застосунком, що взаємодіє з базою даних.
База даних зберігається на віддаленому сервері.

3.4 Технічні засоби, що використовуються

Програмне забезпечення можливо використовувати н будь-якій обчислювальній техніці, де є браузер загального використання(Опера, Гугл браузер).

Середні вимоги для браузера

- Пам’ять: 512 Mb;
- Здатність екрану > 800 х 600 пікселів.

3.5 Вхідні дані

Вхідними даними є дані, що водяться користувачем.
Усі дані зберігаються на сервері, та не зберігаються локально на комп’ютері користувача.

3.6 Вихідні дані

Вихідними даними є сформовані та отримані пакети даних, що повертаються до браузера користувача.

ДОДАТОК Г - ІНСТРУКЦІЯ КОРИСТУВАЧА

Програмне забезпечення у вигляді веб-застосунку для автоматизації процесу тестування та діагностики рівня професійних знань серед здобувачів вищої освіти націлено на поліпшення процесу оцінювання та покращення взаємодії між викладачами та студентами. Для запуску та використання веб-застосунку необхідно мати ПК, ноутбук, планшет або телефон що підтримує роботу сучасних браузерів.

Запуск веб-застосунку відбувається при переході на адресну сторінку.

Першою сторінкою, що зустрічає користувача – є сторінка авторизації. Сторінка авторизації зображена на рис. 1

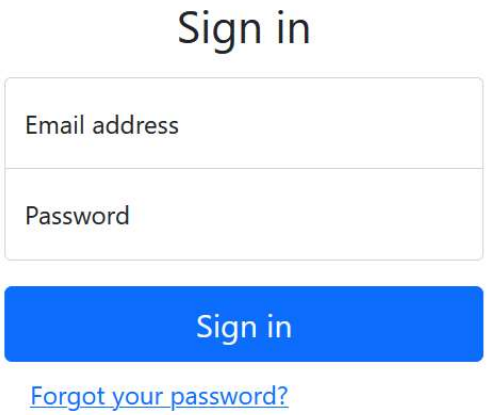

Рисунок Г1 – Сторінка авторизації

У відповідні поля необхідно ввести логін(пошту користувача) та пароль. Якщо введені логін і пароль вірні, відбувається авторизація у системі згідно ролі що закріплена за логіном користувача у базі даних. У іншому випадку відображається повідомлення про відповідну помилку.

1 Адміністратор

1.1 Перегляд загального списку тестів

Для того, щоб переглянути загальний список тестів, необхідно увійти в систему використовуючи облікові дані адміністратора та перейти з головної сторінки на сторінку списку тестів. Список тестів адміністратора представлено на рис. 2

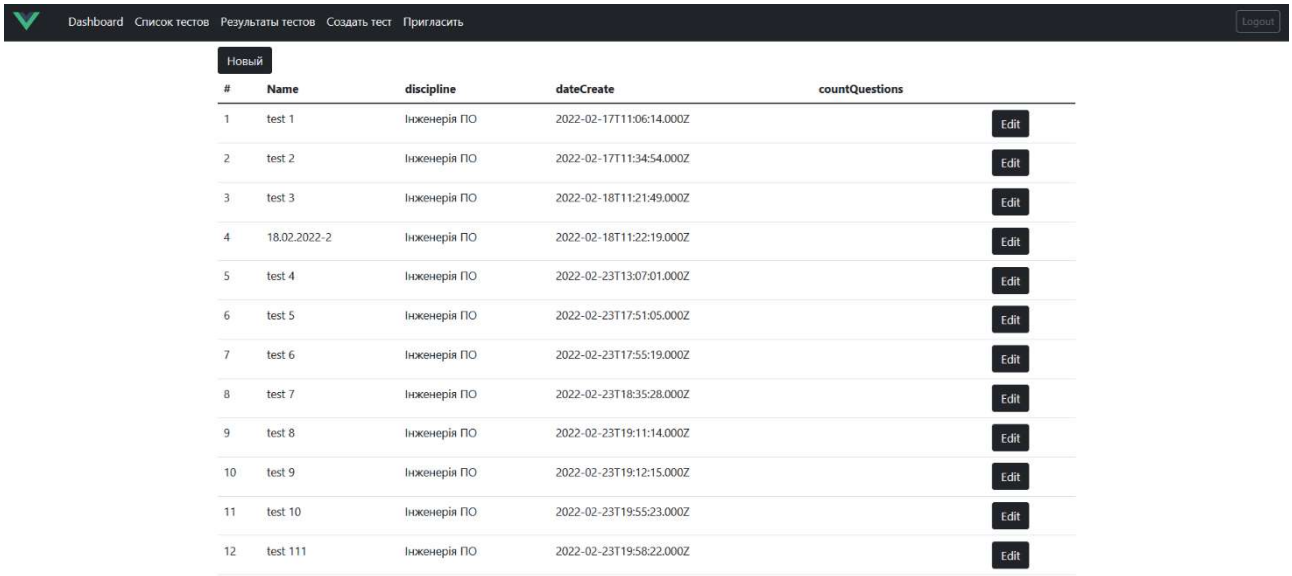

Рисунок Г2 – Сторінка списку тестів

1.2Редагування тесту

Для того, щоб редагувати тест, необхідно перейти на сторінку списку тестів, обрати необхідний тест та натиснути кнопку “Edit”, після цього користувач буде переправлений на сторінку редагування тесту. Сторінку редагування тесту представлено на рис. 3.

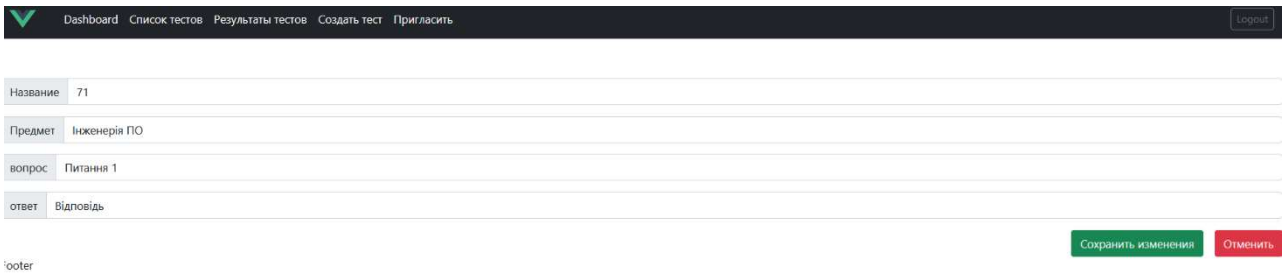

Рисунок Г3 – Сторінка редагування тесту

2 Студент

2.1 Проходження тесту

Для того, щоб переглянути загальний список тестів, необхідно увійти в систему використовуючи облікові дані студента та перейти з головної сторінки на сторінку списку тестів. Обрати необхідний тест та запустити після цього користувач буде перенаправлений на сторінку проходження тесту. Сторінку проходження тесту представлено на рис. 4.

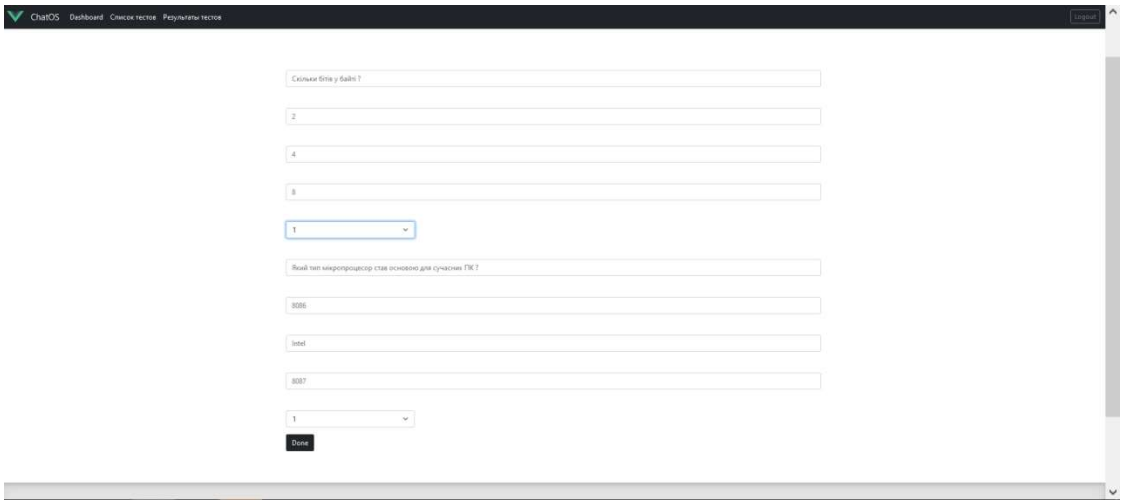

Рисунок Г4 – Сторінка проходження тесту

ДОДАТОК Д - ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

1 Об'єкт випробування

Об’єктом випробування є програмне забезпечення у вигляді веб-застосунку для автоматизації процесу тестування та діагностики рівня професійних знань серед здобувачів вищої освіти, який було розроблено в рамках даної кваліфікаційної роботи.

2 Мета випробування

Метою випробування є перевірка заявлених характеристик, рівня працездатності та виявлення можливих недоліків.

3 Вимоги до веб-застосунку

При проведенні випробувань функціональні характеристики програми перевіряються на відповідність вимогам, наведених у пункті «Вимоги до функціональних характеристик».

4 Вимоги до програмної документації

Програмна документація повинна включати в себе наступні документи:

- Текст програми;
- опис програми;
- інструкція користувача.

5 Засоби і порядок випробувань

5.1 Технічні засоби, що використовуються під час випробувань

Склад технічних засобів, що використовувалися під час випробувань:

- Процесор: x86;
- Пам'ять: 8Гб;
- Здатність екрану: 800 x 600 пікселів;

5.2 Програмні засоби, що використовуються під час випробувань

Склад програмних засобів, що використовувалися під час випробувань: Браузери:

- Google Chrome;
- Opera.

5.3 Порядок проведення випробувань

Порядок проведення випробувань складається з перевірки вимог до функціональних характеристик програмного продукту та перевірки вимог до програмної документації.

6 Методи випробувань

6.1 Методика проведення перевірки вимог до програмної документації

Перевірка дотримання вимог програмної документації на програмний продукт виконується візуально. У ході перевірки зіставляється склад і комплектність програмної документації, представленої розробником, з переліком програмної документації, наведеним у пункті «Склад програмної

документації, пропонованої на випробування» цього документу. Перевірка вважається завершеною у випадку відповідності складу та комплектності програмної документації, представленої розробником, переліком програмної документації, наведеному у зазначеному вище пункті.

6.2 Методика проведення перевірки вимог до функціональних характеристик програмного продукту

Перевірка працездатності програми виконується згідно з пунктом«Вимоги до функціональних характеристик» постановки задачі. Перевірка вважається завершеною у разі відповідності складу і послідовності дій, при виконанні даної перевірки.
В процесі тестування була перевірена функціональність програмного забезпечення.
У табл. 1 наведено перелік випробувань основних функціональних можливостей.

Таблиця 1 – Методика випробування

No	Дія	Очікуваний результат	Результат	Зауваження
1	Перевірка правильності переходу на головну сторінку адміністратора	На екрані відображається головна сторінка адміністратора	Виконано	
2	Перевірка правильності відображення загального списку тестів	На екрані відображається список усіх тестів	Виконано	
3	Перевірка правильності редактування тестів	На екрані відображається сторінка редагування тесту	Виконано	
4	Перевірка правильності переходу на головну сторінку студента	На екрані відображається головна сторінка адміністратора студента	Виконано	
5	Перевірка правильності проходженя тесту	Програма повинна запутити проходження тесту за зберігти результати	Виконано	

Для кожної дії необхідно провести випробування та перевірити правильність виконання описаних функцій.