Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ ПЕРЕВІРКИ ЗНАНЬ
DEVELOPMENT OF SOFTWARE FOR AUTOMATED KNOWLEDGE ASSESSMENT

Виконав(ла): студент(ка) 4 курсу, групи КНТ-122
Спеціальності 121 Інженерія програмного забезпечення
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

КАМЄНЄВ Р.О.
(ПРІЗВИЩЕ та ініціали)
Керівник СТЕПАНЕНКО О.О.
(ПРІЗВИЩЕ та ініціали)
Рецензент КОРОТКИЙ О.В.
(ПРІЗВИЩЕ та ініціали)

2026

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
_____С.О. СУББОТІН
«____» _____2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

КАМЄНЄВА Руслана Олександровича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розроблення програмного забезпечення для автоматизації перевірки знань. Development of Software for Automated Knowledge Assessment

керівник проєкту (роботи) к.т.н., доцент Степаненко Олександр Олексійович,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом закладу вищої освіти від " 07 " квітня 2026 року № 139
2. Строк подання студентом проєкту (роботи) 20 травня 2026 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СТЕПАНЕНКО О.О., доцент		
Нормоконтроль	БЄЛОВА А.В., асистент		

7. Дата видачі завдання « 07 » квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки.		
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження	5 тиждень	Розділ 4
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів	6 тиждень	Додатки
	до неї.		
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка) _____ Руслан КАМЄНЄВ
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Олександр СТЕПАНЕНКО
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 88 с., 42 рис., 6 табл., 4 дод., 18 джерел.

ПЕРЕВІРКА ЗНАНЬ, JAVASCRIPT, REACT.JS, ЗАСТОСУНОК, МОВА ПРОГРАМУВАННЯ, МОДЕЛЬ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об’єкт дослідження – процес розробки програмного забезпечення для автоматизації перевірки знань.

Предмет дослідження – програмні засоби для автоматизації перевірки знань.

Мета роботи – дослідження та розробка програмного забезпечення для автоматизації перевірки знань з метою покращення ефективності контролю знань, зменшення витрат часу та забезпечення прозорості оцінювання.

Матеріали, методи та технічні засоби: мова програмування JavaScript, фреймворк React.js.

Результати. Розроблено програмне забезпечення для автоматизації перевірки знань за допомогою мови програмування JavaScript та фреймворку React.js.

Практична цінність роботи полягає у програмній реалізації застосунку для автоматизації перевірки знань.

Висновки. Реалізовано застосунок для створення віддалених завдань у вигляді WEB-застосунку.

Виконано проєктування програмного забезпечення для автоматизації перевірки знань. Розроблено програмне забезпечення для автоматизації перевірки знань. Здійснено тестування розробленого програмного забезпечення для автоматизації перевірки знань.

Галузь використання – перевірка знань.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 88 p., 42 figures, 6 tables, 4 appendices, 18 sources.

KNOWLEDGE TESTING, JAVASCRIPT, REACT.JS, APPLICATION, PROGRAMMING LANGUAGE, MODEL, SOFTWARE.

The object of the study is the process of developing software for automating knowledge testing.

The subject of the study is software tools for automating knowledge testing.

The purpose of the work is research and development of software for the automation of knowledge verification in order to improve the efficiency of knowledge control, reduce time consumption and ensure transparency of evaluation.

Materials, methods and technical means: JavaScript programming language, React.js framework.

Results. Software for automating knowledge testing has been developed using the JavaScript programming language and the React.js framework.

The practical value of the work lies in the software implementation of the application for automating knowledge testing.

Conclusions. An application for creating remote tasks in the form of a WEB application has been implemented.

Software design for automating knowledge testing has been completed. Software for automating knowledge testing has been developed. Testing of the developed software for automating knowledge testing has been carried out.

Field of use – knowledge testing.

ЗМІСТ

С.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

JS – JavaScript;

SPA – Single Page Application;

БД – база даних;

ОС – операційна система;

ПЗ – програмне забезпечення.

ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується активним впровадженням програмних рішень у різні сфери діяльності, зокрема в освітній процес. Одним із ключових компонентів навчання є перевірка знань, яка забезпечує оцінювання рівня підготовки та формування зворотного зв’язку між учасниками навчання. В умовах зростання обсягів навчальної інформації, поширення дистанційного та змішаного навчання виникає потреба у використанні сучасних програмних засобів, здатних автоматизувати процеси перевірки знань та підвищити їх ефективність [1].

Традиційні підходи до контролю знань часто супроводжуються значними витратами часу, складністю обробки результатів і впливом суб’єктивного чинника. У свою чергу, автоматизація перевірки знань за допомогою програмного забезпечення дозволяє забезпечити швидке отримання результатів, підвищити об’єктивність оцінювання, а також створити зручні умови для організації навчального процесу в онлайн-середовищі. Особливо актуальним є використання вебзастосунків, які не потребують складного встановлення та можуть бути доступними з різних пристроїв [1].

Актуальність даної дипломної роботи зумовлена необхідністю розробки сучасного програмного забезпечення для автоматизації перевірки знань, яке відповідало б вимогам зручності, функціональності та надійності. Використання сучасних мов програмування та фреймворків, зокрема JavaScript і React.js, створює можливості для реалізації інтерактивних вебзастосунків із розширеним функціоналом та інтуїтивно зрозумілим інтерфейсом користувача.

Об’єктом дослідження є процес розробки програмного забезпечення для автоматизації перевірки знань.

Предметом дослідження є програмні засоби для автоматизації перевірки знань.

Метою дипломної роботи є дослідження та розробка програмного забезпечення для автоматизації перевірки знань. Для досягнення поставленої мети в роботі передбачається вирішення таких основних завдань:

– аналіз сучасних підходів та існуючих програмних засобів автоматизації перевірки знань;

– визначення вимог до функціоналу програмного забезпечення для перевірки знань;

– здійснення проєктування програмного забезпечення для автоматизації перевірки знань;

– реалізувати вебзастосунок для автоматизації перевірки знань з використанням мови програмування JavaScript та фреймворку React.js;

– проведення тестування розробленого програмного забезпечення та оцінка його працездатність.

У процесі виконання дипломної роботи використано такі матеріали, методи та технічні засоби: мову програмування JavaScript, фреймворк React.js, методи аналізу та порівняння, проєктування програмного забезпечення, програмну реалізацію та тестування. Обрані інструменти дозволяють створювати сучасні вебзастосунки з високим рівнем інтерактивності та зручності для користувачів.

Структурно дипломна робота складається зі вступу, кількох розділів, у яких розглядаються теоретичні аспекти автоматизації перевірки знань, проєктування та реалізація програмного забезпечення, результати тестування, а також висновків, у яких узагальнено отримані результати та визначено напрями подальшого вдосконалення розробленого програмного продукту.

Дослідження цієї теми спрямоване на розуміння основних вимог до подібних систем, вибір оптимальних технологій для реалізації, проєктування зручного та функціонального інтерфейсу, а також створення надійного веб-застосунку, який відповідатиме сучасним вимогам користувачів. Результатом такої роботи є програмний продукт, що має практичне значення і потенціал до подальшого розвитку в умовах зростаючої цифровізації суспільства.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття віддаленого тестування

Кілька десятиліть тому проведення випробувань передбачало, що фахівець запрошував учасників виконувати завдання на папері, або на стаціонарному комп'ютері, який не має доступу до мережі, а потім спостерігав і фіксував їхні результати. Не існувало такого поняття, як «очні» чи «віддалені» випробування, оскільки всі мали перебувати в одній кімнаті, щоб проводити будь-яке дослідження.

Але відтоді неперервний потік технологічних новацій та нових інструментів випробувань зробив дистанційне тестування здійсненним варіантом. Сьогодні дистанційне тестування не є рідкістю, такому виду випробувань зазвичай надається перевага [2]-[3].

Дистанційне тестування – це будь-яке випробування, яке відбувається, коли учасник і дослідник знаходяться в різних місцях. Дистанційне випробування набуло поширення на практиці з розвитком технологічних новацій і проводиться за допомогою онлайн-інструментів. Сеанси зазвичай відбуваються через платформу для тестування, яка записує, як люди проходять випробування, збирає дані та генерує висновки, які можна негайно застосувати на практиці [2]-[3].

Використання віддаленого підходу до випробувань має деякі великі переваги:

– знайти бажаючих учасників набагато легше, оскільки користувачі можуть проходити випробування, коли завгодно і де завгодно;

– швидше та дешевше проводити кількісне випробування, тому що є більше шансів отримати статистично значущі результати;

– можна перевірити свій дизайн на ширшому колі людей з різних куточків світу;

– час між створенням випробування та отриманням результатів набагато коротший, а це означає, що можна дуже швидко отримати аналітичні дані [2]-[3].

Деякі типові переваги, які люди бачать у віддаленому випробуванні, полягають у ширшій та часто різноманітнішій аудиторії для опитування, а не лише в людях, які знаходяться поруч. Також менше накладних витрат пов'язано з проведенням дослідження, оскільки для нього, як правило, потрібен лише пристрій та підключення до Інтернету.

Дистанційні випробування легко організувати, оскільки учасники можуть проходити їх коли завгодно та де завгодно. Можна поділитися посиланням на випробування у соціальних мережах, або в електронній розсилці та отримати корисний відгук за лічені хвилини. Це величезна перевага порівняно з очним випробуванням, оскільки можна перевірити набагато більше людей за набагато менший час і з набагато меншими зусиллями [2]-[3].

Дистанційне тестування користувачів також добре поєднується з кількісним випробуванням, оскільки воно дозволяє отримувати дані з великої вибірки учасників випробувань. Можете легко вимірювати ефективність свого випробування, або відстежувати показники зручності використання, оскільки дані можна записувати автоматично, без клопоту з особистими сеансами.

Також існує аргумент, що люди поводяться більш реалістично, коли складають випробування у своєму природному середовищі, тому висновки будуть базуватися на сценарії, ближчому до реального життя [2]-[3].

Недоліком віддалених випробувань зручності використання є менший контроль над тестовим середовищем та процедурою загалом. Наприклад, під час проведення віддаленого немодерованого тестування, якщо завдання незрозуміле, ви не можете бути присутніми, щоб детальніше пояснити його та спрямувати у правильному напрямку;

Щоб мінімізувати ці ризики, потрібно розробити опис випробування та рекомендації на початку тесту, щоб учасники знали, чого очікувати. Якщо

проводити модерований дистанційний тест, потрібно повідомитт учасника про мету сеансу та переконайтеся, що він має всю необхідну інформацію для розуміння контексту [2]-[3].

Зрештою, вкрай важливо провести пілотне випробування немодерованих віддалених тестів, щоб переконатися, що спосіб написання завдань є кришталево зрозумілим. Віддалене тестування не менш трудомістке, ніж очне випробування, воно часто вимагає більше планування та комунікації.

1.2 Основні відмінності он-лайн та особового тестування

Спалах COVID-19 створив нові можливості в сферах освіти та випробувань. Вперше в сучасній пам'яті освіту та випробування довелося проводити поза звичними приміщеннями. Раптом студенти всіх типів не змогли з'явитися до аудиторії чи тестового осередку для складання іспитів. Це було складно і заплутано для багатьох. Здавалося дивним і незручним виходити за рамки звичайного розпорядку. Однак з часом ці зміни почали відчуватися комфортніше, і зрештою багато хто віддав перевагу дистанційним заняттям та випробуванням [4].

Останнім часом спостерігається прагнення повернутися до способу тестування та навчання, який ми використовували до пандемії. Заклики до припинення дистанційного навчання та тестування зростають, водночас зростають заклики до розширення пропозицій дистанційного навчання. Який варіант кращий? Що ж, на це питання немає однозначної простої відповіді. Існує кілька чинників, які враховуються при визначенні того, чи є очне випробування кращим вибором, ніж дистанційне випробування, і навпаки [4].

Розглядаючи, який варіант може бути найкращим, важливо бути добре поінформованим про свій вибір та зважити всі «за» та «проти» кожного з них.

Передумови якісного он-лайн-тестування:

– по-перше, важливо визначити міцність та стабільність інтернет-з’єднання. Оскільки онлайн-тести можуть тривати від 30 хвилин до трьох-чотирьох годин, вкрай важливо, щоб Інтернет був надійним та стійким;

– по-друге, потрібно створити умови для приватності і місце для проведення іспиту. Несподіваний вхід когось може відволікти, і це може призвести до втрати дорогоцінного часу через необхідність перерви;

– по-третє, потрібен комп'ютер, якому буде не більше трьох-чотирьох років. Це гарантує, що екрани, клавіатури, камери, мікрофони та динаміки зможуть працювати належним чином із програмним забезпеченням, веб-сайтами та платформами, що використовуються у вашому випробуванні.

Позитивні сторони дистанційного тестування включають:

Більш зручний графік:

– підходить для людей із тривогою перед іспитами;

– підходить для тих, хто мешкає занадто далеко від центрів випробувань;

– забезпечує можливість тестування для людей з інвалідністю або проблемами з мобільністю;

– комфортніше бути вдома або в зручному місці.

Недоліки, пов'язаних з дистанційним тестуванням, зокрема:

– проблеми шахрайства або нечесності;

– не всі тести доступні для дистанційного складання;

– якщо у немає сучасного комп'ютера, або високошвидкісного Інтернету, можуть виникнути проблеми;

– з комп'ютером можуть траплятися збої;

– забезпечення конфіденційності та контроль переривань може бути складним завданням [4].

Онлайн-тестування пропонує кандидатам гнучкість у зручніший час, комфорт випробування вдома та можливість проходити тест без додаткових турбот про поїздки чи витрати. Для кандидатів, які проживають у більш

віддалених районах, це може означати, що їм не доведеться витрачати години на поїздки до та з місця проведення тестування.

Речі, які слід брати до уваги під час очного тестування. Багато людей надають перевагу цьому методу випробування, і є деякі ситуації, які вимагають особової явки кандидата. Наприклад, для тих, хто працює в галузях косметології та перукарства, деякі ліцензійні ради вимагають особистого проведення практичного тесту для оцінки навичок.

Переваги особового тестування включають:

– менше проблем, пов'язаних з шахрайством та чесністю;

– оточення може допомогти покращити концентрацію;

– проблеми з обладнанням можуть вирішити співробітники тестового осередку;

– інспектори можуть зменшити турботи, пов'язані з шумом, оточенням або перешкодами.

Недоліки особового тестування, зокрема:

– через подорожі, паркування, тощо, особова присутність може зайняти багато часу;

– якщо доводиться платити за проїзд, використовувати бензин для керування автомобілем, або платити за паркування, особове тестування може бути дорогим;

– планування особового тестування вимагає додаткового планування та пропонує меншу гнучкість;

– якщо обрати варіант особового іспиту, потрібно планувати прибути на місце проведення іспиту завчасно, маючи достатньо часу, щоб знайти паркувальне місце, зареєструватися та влаштуватися до того, як вас запросять до зали. Слід прибути приблизно за 30 хвилин до початку, щоб можна було зробити останній огляд, або розслабитися, глибоко подихати [4].

Який з методів тестування кращий? Відповідь на це питання однозначна – залежить від обставин. Якщо потрібно буде розглянути власні обставини, щоб вирішити, який варіант найкращий. Чи доступний потрібний

іспит для дистанційного тестування, чи вам потрібно складати його особисто? Чи є поблизу тестовий осередок, чи потрібно буде подорожувати? Чи можете ви гарантувати середовище без відволікаючих чинників з хорошим підключенням до Інтернету, чи потрібні технічні ресурси, розташовані в осередку? Після чесної оцінки ситуації можна визначити, який варіант найкраще підійде.

1.3 Поняття програмного забезпечення для створення тестів

Сфера освіти все більше цифровізується, і пошук підходящого програмного забезпечення для створення тестів є важливим як для викладачів, так і для навчальних закладів.

Програмне забезпечення для створення тестів – це потужний освітній інструмент, розроблений для спрощення процесу формування, проведення та аналізу тестів і оцінювань. Ці цифрові платформи дозволяють викладачам створювати широкий спектр вікторин, іспитів та опитувань з використанням різних форматів запитань, таких як питання з вибором однієї правильної відповіді, есе та інтерактивні елементи. Завдяки інтуїтивно зрозумілим інтерфейсам викладачі можуть налаштовувати оцінювання відповідно до конкретних навчальних цілей та рівнів учнів [5].

Крім того, ці програмні рішення часто включають такі функції, як автоматичне оцінювання, зворотний зв'язок у режимі реального часу та детальна аналітика успішності, що надає безцінну інформацію про поступ учнів. Підвищуючи ефективність, сприяючи інтерактивності та пропонуючи комплексні можливості оцінювання, програмне забезпечення для створення тестів революціонізує спосіб, у який викладачі розробляють та керують оцінюваннями в сучасних класах та онлайн-середовищі навчання.

Он-лайн конструктори тестів вирізняються зручними інтерфейсами, що дозволяє викладачам, незалежно від технічних знань, легко орієнтуватися та створювати оцінювання. Інтуїтивно зрозумілі функції, прості пункти меню та

чіткі інструкції спрощують процес створення різноманітних та інтерактивних тестів. Ця простота використання значно скорочує час навчання, дозволяючи викладачам зосередитися на розробці цікавого контенту, а не на боротьбі зі складними програмними механізмами [5].

Використовуючи можливості хмарних технологій, сучасне програмне забезпечення для створення тестів безперебійно працює он-лайн. Цей хмарний підхід пропонує гнучкість, дозволяючи викладачам отримувати доступ до своїх оцінювань з будь-якого пристрою з підключенням до Інтернету. Він усуває обмеження фізичного сховища, гарантуючи, що викладачі можуть створювати, змінювати та адмініструвати тести з будь-якого місця, підвищуючи продуктивність та сприяючи співпраці між викладацькими командами.

Провідне програмне забезпечення для створення тестів не обмежується лише основними типами питань. Воно надає низку розширених функцій, включаючи інтеграцію мультимедіа, автоматизовані алгоритми оцінювання та аналітику в режимі реального часу. Ці функції підвищують глибину та інтерактивність оцінювань, враховуючи різні стилі навчання та сприяючи всебічному розумінню предметів серед учнів [5].

Програмне забезпечення для створення тестів дозволяє розширювати конфігурації, дозволяючи викладачам налаштовувати оцінювання відповідно до своїх конкретних вимог. Від встановлення часових обмежень до рандомізації питань, викладачі можуть точно налаштовувати тести, щоб створювати унікальні версії для різних учнів, забезпечуючи справедливість та чесність оцінювання.

В епоху мобільного навчання сумісність зі смартфонами та планшетами має першорядне значення. Найкраще програмне забезпечення для створення тестів забезпечує адаптивний дизайн, що гарантує безперешкодний доступ та оптимальний користувацький досвід на безлічі пристроїв. Ця сумісність з мобільними пристроями сприяє доступності та гнучкості, враховуючи різноманітні вподобання та навчальні звички сучасних учнів [5].

1.4 Критерії вибору програмного забезпечення для створення тестів

Нюанси вибору програмного забезпечення для створення тестів:

– зручність використання. Потрібно обирати інтуїтивно зрозумілі інтерфейси та функцію. Зручні платформи скорочують час навчання, дозволяючи викладачам зосередитися на формуванні контенту, а не на складнощах програмного забезпечення;

– функції. Потрібно обирати програмне забезпечення, що пропонує різноманітні типи запитань, інтеграцію мультимедіа, автоматичне оцінювання та зворотний зв'язок у режимі реального часу. Розширені функції покращують оцінювання, враховуючи різні стилі навчання та сприяючи захопливому навчальному досвіду;

– конфігурації. Потрібно обирати програмне забезпечення, що дозволяє розширити конфігурації. Налаштовуванні параметри, такі як обмеження часу, рандомізація запитань та адаптивне тестування, забезпечують індивідуальне оцінювання, враховуючи індивідуальні потреби студентів та підтримуючи цілісність тестування;

– спільний доступ. Платформи, які сприяють легкому обміну. Потрібно обирати варіанти, що дозволяють безпечно розповсюджувати тести електронною поштою, за унікальними посиланнями або інтеграцією із системами управління навчанням. Можливості обміну мають бути безперебійними та ефективними, забезпечуючи швидкий доступ для студентів;

– ціноутворення. Оцініть структуру ціноутворення та масштабованість. Виберіть програмне забезпечення, що пропонує гнучкі плани, що враховують розмір та бюджет навчального закладу. Враховуйте такі фактори, як кількість користувачів, ємність сховища та доступні функції, під час оцінки економічної ефективності платформи [6].

Враховуючи ці аспекти, викладачі можуть приймати обґрунтовані рішення, вибираючи програмне забезпечення для створення тестів, яке ідеально відповідає їхнім навчальним цілям, підвищує залученість студентів та спрощує процес оцінювання для створення більш продуктивного та інтерактивного навчального середовища.

1.5 Порівняння програмного забезпечення для створення тестів

У сучасному світі вибір потрібного програмного забезпечення для створення тестів є ключовим. Виділяється декілька цікавих варіантів програмного забезпечення для створення тестів (рисунок 1.1). ExamSoft надає надійні функції, наприклад безпечне адміністрування іспитів та поглиблена аналітика. ProProfs вирізняється зручністю використання та різноманітністю типів запитань. Respondus бездоганно зливається з системи управління навчанням та забезпечує безпечне середовище для тестування. Quizizz відрізняється своїм гейміфікованим підходом, що підвищує залученість студентів. Тим часом ClassMarker пропонує масштабованість, дозволяючи викладачам ефективно створювати та оцінювати тести. Завдяки своїм комплексним функціям, включаючи мультимедійні запитання та детальну аналітику, QuizMaker є ще одним чудовим вибором. ThinkExam гарантує безпечне он-лайн оцінювання та налаштовувані звіти [6].

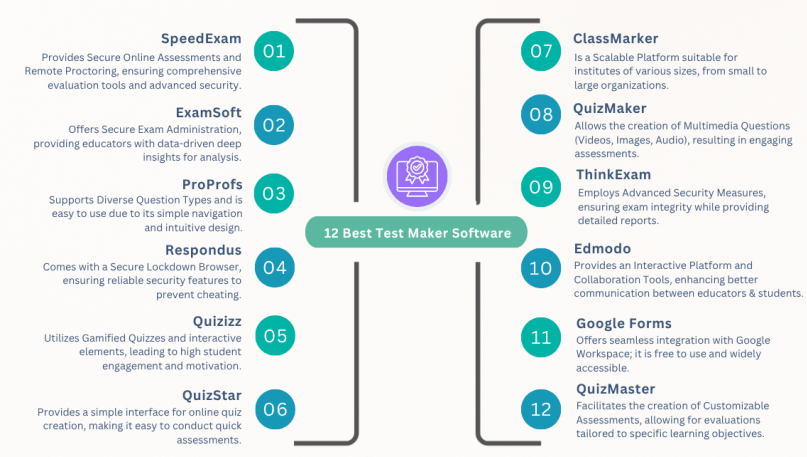

Рисунок 1.1 – Програмне забезпечення для створення тестів [6]

Крім того, Edmodo надає інтерактивну платформу як для викладачів, так і для студентів, покращуючи співпрацю та комунікацію. Кожен із цих програмних варіантів задовольняє унікальним стилям викладання та потребам учнів, забезпечуючи динамічний та ефективний процес тестування [6].

Переваги:

– комплексні інструменти оцінювання;
– зручний інтерфейс;
– розширені заходи безпеки;
– інтерактивне оцінювання;
– детальна аналітика.

Недоліки:

– обмежені можливості безкоштовної версії;
– потреба навчання для широкого використання;
– складність початкового налаштування;
– обмежена інтеграція з іншими платформами;
– обмежений вибір типів питань.

Огляд програмного забезпечення для створення тестів «ExamSoft» [6].

Особливості:

– безпечне адміністрування іспитів;
– розширена аналітика;
– безшовна інтеграція з lms;
– налаштовувані параметри;
– зворотній зв'язок та оцінювання в режимі реального часу.

Переваги:

– аналітика на основі даних для освітян;
– безпечне середовище для тестування;
– ефективна система оцінювання;
– оптимізовані можливості інтеграції;
– варіанти адаптивного тестування.

Недоліки:

– складність початкового налаштування;

– крива навчання для складних функцій;

– обмежені варіанти типів питань;

– потрібне стабільне підключення до інтернету;

– преміум-функції платні;

Огляд програмного забезпечення для створення тестів «ProProfs» [6].

Особливості:

– різноманітні типи питань (питання з множинним вибором, есе, вікторини);

– зручний інтерфейс для викладачів та учнів;

– інтерактивні елементи (зображення, відео);

– налаштовувані вікторини та опитування;

– безшовна інтеграція з платформами lms.

Переваги:

– проста навігація та інтуїтивно зрозумілий дизайн;

– велика бібліотека питань для різноманітних оцінювань;

– інтерактивні елементи;

– зворотній зв'язок у режимі реального часу для негайного аналізу;

– прості варіанти співпраці та обміну.

Недоліки:

– обмежені можливості безкоштовної версії;

– періодичні збої або затримки інтерфейсу;

– деякі розширені функції можуть вимагати певного часу навчання;

– обмежена інтеграція зі сторонніми інструментами;

– час відповіді служби підтримки клієнтів може відрізнятися.

Огляд програмного забезпечення для створення тестів «Respondus» [6].

Особливості:

– безшовна сполучність із різними платформами lms;

– захищене блокування браузера для екзаменаційного середовища;

– налаштовувані параметри для різноманітних оцінювань;

– докладна аналітика та звітність щодо іспитів;

– сумісність із багатьма ґаджетами та операційними системами.

Переваги:

– надійні функції безпеки запобігають обману;

– зручна система оцінювання заощаджує час викладачам;

– гнучкі формати запитань та відповідей;

– зручний інтерфейс як для викладачів, так і для учнів;

– проста інтеграція з популярними платформами lms;

Недоліки:

– обмежена різноманітність зразків запитань;

– проблеми сумісності з певними браузерами;

– складність початкового налаштування та конфігурації;

– обмежені інтерактивні складники в оцінюваннях;

– потрібне стабільне інтернет-з'єднання.

Огляд програмного забезпечення для створення тестів «Quizizz» [6].

Особливості:

– гейміфіковані вікторини та інтерактивні складники;

– зворотній зв'язок у режимі реального часу та відслідковування ефективності;

– різноманітні типи запитань (питання з множинним вибором, «правда/неправда», відкриті);

– можливості сполучення з різними навчальними платформами;

– навчання в темпі студента задля індивідуального досвіду.

Переваги:

– висока зацікавленість студентів;

– цікавий навчальний досвід з ігровими функціями;

– детальна аналітика ефективності для викладачів;

– прості варіанти спільного доступу та співпраці;

– доступність на різних ґаджетах та браузерах.

Недоліки:

– обмежена складність питань для предметів вищого рівня;

– періодичні затримки сервера під час найбільшого навантаження;

– обмежений нагляд над середовищем іспиту для викладачів;

– базові аналітичні функції порівняно з деякими опонентами;

– складність для викладачів, які вперше працюють із гейміфікованими платформами.

Огляд програмного забезпечення для створення тестів «ClassMarker» [6].

Особливості:

– масштабований для закладів різного розміру;

– різноманітні типи питань (питання з множинним вибором, есе, завантаження файлів);

– налаштовувані параметри оцінювання та зворотного зв'язку;

– захищене середовище для іспитів з онлайн-наглядом;

– можливості інтеграції з популярними платформами lms.

Переваги:

– підходить як для невеликих закладів, так і для великих організацій;

– інтуїтивно зрозумілий інтерфейс для викладачів та учнів;

– зручна система оцінювання з автоматизованими функціями;

– захищені варіанти онлайн-прокторингу;

– співпраця з провідними платформами lms для отримання досвіду.

Недоліки:

– обмежені розширені функції порівняно з деякими опонентами;

– може бракувати деяких інтерактивних складників, що є на інших платформах;

– складність для викладачів, які вперше користуються платформою;

– періодичні проблеми сумісності з певними ґаджетами/браузерами;

– обмежені можливості налагодження.

Огляд програмного забезпечення для створення тестів «QuizMaker» [6].

Особливості:

– мультимедійні питання (відео, зображення, аудіо);

– докладні інструменти аналітики та звітності;

– налаштовувані вікторини та оцінювання;

– інтерактивний навчальний досвід;

– можливості сполучення з різними платформами.

Переваги:

– цікаві оцінювання з мультимедійними складниками;

– глибоке розуміння успішності учнів;

– інтерактивний навчальний досвід підвищує зацікавленість;

– зручний дизайн для викладачів.

Недоліки:

– складність початкового налаштування та конфігурації;

– для впевненого використання може знадобитися навчання;

– обмежені можливості безкоштовної версії;

– деякі розширені функції можуть вимагати додаткової оплати;

– періодичні оновлення можуть порушувати роботу наявних робочих процесів.

Огляд програмного забезпечення для створення тестів «ThinkExam» [6].

Особливості:

– захищена оцінка тестів за допомогою розширених заходів безпеки;

– налаштовувані звіти та відслідковування у режимі реального часу;

– різноманітні типи запитань (питання з множинним вибором, описові, кодування);

– проста інтеграція з популярними платформами lms;

– варіанти дистанційного нагляду за он-лайн іспитами.

Переваги:

– надійні заходи безпеки забезпечують цілісність іспиту;

– докладні звіти про ефективність допомагають викладачам в аналізі;

– гнучкість у типах питань для різноманітних оцінок;

– проста інтеграція з популярними платформами lms;

– опції дистанційного нагляду підвищують безпеку іспиту.

Недоліки:

– обмежені можливості безкоштовної версії;

– потенційна складність, особливо для початківців;

– може знадобитися навчання для широкого використання функцій;

– обмежені інтерактивні складники порівняно з деякими аналогами;

– періодичні оновлення можуть впливати на наявні робочі процеси.

Огляд програмного забезпечення для створення тестів «Edmodo» [6].

Особливості:

– інтерактивна платформа для викладачів та учнів;

– інструменти для співпраці (завдання, тести, обговорення);

– функції зворотного зв'язку та оцінювання у режимі реального часу;

– інтеграція з освітніми програмами та контентом;

– захищене спілкування та можливості обміну даними.

Переваги:

– покращена комунікація між викладачами та учнями;

– цікаві та інтерактивні навчальні матеріали;

– зворотній зв'язок у режимі реального часу сприяє негайному покращенню;

– захищені варіанти обміну та співпраці;

– інтеграція із різноманітними освітніми інструментами.

Недоліки:

– для обговорень може знадобитися активна модерація;

– обмежені розширені функції оцінювання порівняно зі спеціалізованими платформами;

– складність використання для викладачів, які вперше користуються інтерфейсом;

– періодичні затримки сервера під час найбільшого навантаження;

– обмежені можливості виставлення оцінок.

Огляд програмного забезпечення для створення тестів «Google Forms» [6].

Особливості:

– зручний інтерфейс для створення опитувань та вікторин;

– сполучність з інструментами Google Workspace (документи, таблиці);

– інтеграція та можливості обміну даними у режимі реального часу;

– базові аналітичні функції для отримання аналітичних даних;

– доступно на різних ґаджетах та браузерах.

Переваги:

– безкоштовний у використанні та широка доступність для освітян;

– інтеграція з інструментами Google Workspace;

– легкі варіанти інтеграція та обміну інформацією зі студентами;

– базова аналітика для швидкого аналізу відповідей;

– налаштовувані форми та тести для різноманітних оцінювань.

Недоліки:

– обмежені зразки запитань;

– базова аналітика;

– обмежені інтерактивні складники;

– обмежені розширені функції;

– залежить від стабільного інтернету.

Огляд програмного забезпечення для створення тестів «QuizStar» [6].

Особливості:

– створення он-лайн вікторин з простим інтерфейсом;

– базова аналітика для швидкої оцінки;

– прості варіанти обміну та розповсюдження;

– зручна платформа для викладачів та студентів;

– сумісність із різними ґаджетами та браузерами.

Переваги:

– легке створення вікторин за допомогою простих інструментів;

– інтуїтивно зрозумілий інтерфейс для викладачів та учнів;

– прості варіанти спільного використання для ефективного розповсюдження;

– підходить для швидких оцінювань та перевірки знань;

– сумісність із широким спектром ґаджетів та браузерів.

Недоліки:

– обмежена складність питань для поглиблених оцінок;

– базові функції звітності можуть не задовольняти потреби розширених користувачів;

– може бути брак розширених зразків запитань, які можна знайти на спеціалізованих платформах;

– обмежені можливості налагодження оцінок;

– обмежені інтерактивні складники у вікторинах та іспитах.

Огляд програмного забезпечення для створення тестів «QuizMaster» [6].

Особливості:

– налаштовувані оцінки, адаптовані до конкретних потреб;

– докладний зворотний зв'язок та інтерактивні складники;

– можливості сполучення з вибраними платформами;

– зручний дизайн для інтуїтивної навігації;

– захищене середовище для іспитів із заходами проти шахрайства.

Переваги:

– індивідуальні оцінювання для конкретних навчальних цілей;

– інтерактивні запитання;

– зручний дизайн для викладачів та учнів;

– безшовна сполучність із сумісними платформами;

– захищене середовище для іспитів забезпечує справедливе оцінювання.

Недоліки:

– може бракувати деяких розширених функцій, доступних на спеціалізованих платформах;

– обмежена сполучність із певними сторонніми інструментами;

– складність для викладачів, які вперше користуються платформою;

– обмежений вибір зразків питань задля складних оцінювань;

– періодичні оновлення можуть порушувати працю наявних робочих процесів.

Результати аналізу показали, що освітні технології стрімко прогресують. Зараз, як ніколи раніше, освітяни мають безліч варіантів на вибір. Також, можна зробити висновок, що у розглянутого програмного збезпечення є типові недоліки, акутальні для більшості розглянутих платформ і постає актуальна задача для створення такого програмного збезпечення, як б мого зменшити кількість розглянутих недоліків.

1.6 Висновки за розділом 1

У даному розділі було розглянуто теоретичні та практичні аспекти організації тестування знань в умовах цифровізації освітнього процесу. Проаналізовано підходи до проведення випробувань — від традиційного очного тестування до сучасних дистанційних форматів, що стали особливо актуальними внаслідок розвитку інформаційних технологій.

В розділі визначено основні переваги та недоліки дистанційного й очного тестування. Встановлено, що онлайн-тестування забезпечує гнучкість, доступність, економію часу та ресурсів, а також можливість охоплення

ширшої аудиторії. Водночас воно потребує надійного технічного забезпечення, стабільного інтернет-з’єднання та продуманих механізмів контролю доброчесності. Очне тестування, у свою чергу, характеризується вищим рівнем контролю та концентрації, проте є менш гнучким і більш затратним з організаційної точки зору. Зроблено висновок, що вибір форми тестування має здійснюватися з урахуванням конкретних умов, цілей оцінювання та можливостей учасників.

Також у розділі було розкрито поняття програмного забезпечення для створення тестів, його функціональні можливості та роль у сучасній освіті. Визначено ключові критерії вибору таких програмних засобів, зокрема зручність використання, функціональність, налаштовуваність, можливості спільного доступу та економічну доцільність.

Проведено порівняльний аналіз популярних програмних продуктів для створення тестів, у результаті якого виявлено їхні спільні переваги та типові недоліки. Аналіз показав, що, незважаючи на широкий вибір рішень, більшість із них мають обмеження, пов’язані з функціональністю, складністю налаштування, вартістю або інтеграцією з іншими системами.

Отже, результати розділу підтверджують актуальність розробки програмного забезпечення для автоматизації перевірки знань, яке б поєднувало зручність використання, розширений функціонал, надійність і доступність, а також мінімізувало виявлені недоліки існуючих рішень. Отримані висновки є теоретичним підґрунтям для подальшої практичної частини дипломної роботи.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

На сьогоднішній день понад При створенні програмного забезпечення, спрямованого на автоматизацію перевірки знань, одним із критичних етапів є умотивований вибір мови програмування. Від цієї опції залежать такі аспекти, як функціональність системи, її швидкість роботи, простота подальшого супроводу та потенціал для розширення. Враховуючи поставлені завдання та особливості майбутнього застосунку, було прийнято рішення зупинитися на JavaScript як основній мові програмування [7].

На сьогодні JavaScript є однією з найбільш поширених та затребуваних мов у сфері створення веб-застосунків. Його завидну популярність забезпечує, перш за все, багатогранність, оскільки ця мова застосовується як для формування клієнтської частини веб-додатків, так і для втілення серверної логіки. Такий уніфікований підхід дає змогу використовувати єдиний технологічний набір у межах одного проєкту, що помітно спрощує процес розробки та зменшує ризик виникнення помилок [7].

Важливою перевагою JavaScript вважається підтримка асинхронного принципу програмування. Для програмного забезпечення, яке займається автоматизацію перевірки знань, це має першочергове значення, адже система повинна оперативно обробляти результати взаємодії користувачів з системою у режимі реального часу. Завдяки асинхронним механізмам ці операції можуть виконуватися без призупинення основного ланцюжка виконання, що позитивно позначається на швидкодії та зручності експлуатації [7].

JavaScript також вирізняється високим ступенем інтерактивності. Завдяки широким можливостям взаємодії з інтерфейсом користувача стає реальним створення динамічних мап, фільтрів для заходів, інтерактивних переліків та сповіщень. Це особливо цінно для навігаційних систем, де очікується швидка реакція інтерфейсу та інтуїтивно зрозуміла взаємодія з додатком [7].

Значну роль відіграє розвинена сукупність бібліотек та фреймворків. Такі інструменти, як React, Angular та Vue.js, прискорюють розробку фронтенд-частини застосунку, підвищують його надійність та спрощують обслуговування коду. Використання готових рішень дозволяє сфокусуватися на втіленні бізнес-логіки та функціоналу системи, а не на вирішенні типових технічних завдань.

Ще однією вагомою перевагою JS є його кросплатформність. Завдяки сучасним інструментам розробки можливо створювати вебверсії, мобільні програми та навіть десктопні додатки на базі єдиної кодової бази. Це суттєво мінімізує затрати часу й ресурсів, одночасно спрощуючи процес оновлення софту [7].

Масштабна спільнота JavaScript-розробників є додатковим аргументом на користь вибору цієї мови. Інтенсивна підтримка, велика кількість навчальних матеріалів, документації та прикладів реалізації допомагають оперативно знаходити рішення для складних проблем та запроваджувати передові підходи до створення програмних рішень.

Обґрунтування вибору мови програмування для розробки програмного забезпечення для автоматизації перевірки знань наведено у таблиці 2.1 [7]-[9].

Таблиця 2.1 – Обґрунтування вибору мови програмування розробки програмного забезпечення для автоматизації перевірки знань

Критерій порівняння мов програмування	Мова програмування		
	JavaScript	C#	Python
Висока продуктивність виконання програмного коду	+	+–	+–
Придатність і зручність для розробки програмного забезпечення для автоматизації перевірки знань	+	–	–

Кінець таблиці 2.1

Критерій порівняння мов програмування	Мова програмування		
	JavaScript	C#	Python
Розширені можливості інтерактивної взаємодії з користувачем	+	+	+–
Наявність численної та активної спільноти розробників	+	-	+
Підтримка роботи на різних платформах та операційних системах	+	+–	+

Отже, аналіз функціональних вимог, показників продуктивності, потенціалу масштабування та можливостей зв'язку із зовнішніми сервісами випливає, що JavaScript є найбільш доречним вибором для розробки софту для автоматизації перевірки знань.

2.2 Вибір бібліотеки для створення програмного забезпечення автоматизації перевірки знань

При створенні програмного забезпечення інформаційної системи для автоматизації перевірки знань вирішальну роль відіграє вибір бібліотеки для формування користувацького інтерфейсу. Саме він визначає зручність взаємодії користувача із системою, швидкість доступу до функцій та загальне враження від роботи з продуктом. Зважаючи на функціональні потреби, можливості розширення та перспективи подальшого розвитку системи, для реалізації її фронтенд-частини була обрана бібліотека React.

React — це сучасна бібліотека для конструювання інтерфейсів, яка оперує компонентним підходом та використовує мову програмування JavaScript. Основна концепція React полягає у розбитті інтерфейсу на

незалежні, багаторазово застосовні компоненти, кожен з яких відповідає за певну візуальну частину або логіку. Цей підхід суттєво покращує структуру коду, спрощує його обслуговування та полегшує процес додавання нових функціональних можливостей системи [10].

Ключовою перевагою React є застосування віртуального DOM. Замість повної перебудови вебсторінки під час кожної зміни даних, бібліотека аналізує попередній та поточний стани інтерфейсу, оновлюючи лише ті елементи, які зазнали змін. Це дозволяє значно підвищити швидкість роботи вебдодатку, що критично важливо для інформаційної системи творчої студії, де користувачі можуть одночасно оперувати розкладами занять, списками членів, новинами та мультимедійним контентом [10].

React забезпечує ефективне управління станом інтерфейсу, що дозволяє автоматично коригувати відображення даних у відповідь на зміни внутрішніх чи зовнішніх параметрів. Завдяки цьому користувач завжди бачить актуальну інформацію без потреби перезавантажувати сторінку. Цей механізм надзвичайно корисний для динамічних блоків, як-от форми для реєстрації на курси, календарі подій, переліки творчих робіт чи сповіщення [10].

Використання React значно спрощує інтеграцію складних програмних блоків. Компонентна структура дає змогу об'єднувати прості елементи у більш складні архітектури, не порушуючи загальної логіки системи. Це гарантує високу гнучкість при проєктуванні інтерфейсу та дає можливість послідовно розширювати функціонал ПЗ без необхідності радикальних змін у кодовій базі.

Важливою перевагою React є оптимізація супроводу та оновлення програмного забезпечення. Бібліотека має спеціалізовані інструменти, які спрощують перехід між версіями та зменшують ризики появи помилок після апгрейду. Це дає розробникам змогу швидко впроваджувати нові можливості, посилювати безпеку та адаптувати систему до нових запитів користувачів [10].

Окрім того, React володіє розгалуженою екосистемою допоміжних бібліотек та інструментів, що істотно розширюють його базовий функціонал.

Поєднання з HTML та CSS поряд із JavaScript дозволяє автоматизувати обробку подій, керування формами, перевірку даних та відображення складних інтерфейсних елементів. Потоки даних у React організовані таким чином, щоб забезпечити передбачувану поведінку додатка, що позитивно впливає на його стабільність та надійність.

Завдяки своїй гнучкості React дає змогу легко виконувати операції створення, зміни та видалення інтерфейсних компонентів. Це сприяє розробці інтуїтивно зрозумілих і логічно організованих вебдодатків, орієнтованих на кінцевого споживача. Для ПЗ автоматизації перевірки знань це означає можливість формування зручного середовища для адміністраторів та користувачів. Простота React також проявляється у передбачуваній поведінці компонентів і чіткому життєвому циклі. Керування станом у React реалізоване логічно та послідовно, що полегшує роботу з динамічними даними [10].

Бібліотека дозволяє швидко створювати прототипи інтерфейсів без складної початкової конфігурації. React не нав’язує жорсткої структури проєкту, надаючи розробнику свободу вибору архітектурних рішень.Завдяки мінімальній кількості обов’язкових інструментів початкове налаштування проєкту є простим і швидким.

Логіка повторного використання компонентів зменшує обсяг коду та робить його більш зрозумілим. React легко поєднується з HTML і CSS, що дозволяє застосовувати знайомі технології без додаткового навчання. Простота бібліотеки сприяє зменшенню кількості помилок у процесі розробки.

Чітке розділення логіки та представлення підвищує читабельність програмного коду. React дозволяє легко тестувати окремі компоненти без складних налаштувань [10].

Для підтвердження обґрунтованості вибору React було проведено порівняння з іншими бібліотеками, зокрема Vue.js та Preact. Аналіз показав, що хоча згадані рішення також мають свої плюси, React вирізняється потужнішою підтримкою спільноти, ширшими можливостями для

масштабування та наявністю великої кількості навчальної інформації. Результати цього порівняння містяться у таблиці 2.2 [10]-[12].

Таблиця 2.2 – Обґрунтування вибору бібліотеки для побудови інтерфейсів для створення програмного забезпечення автоматизації перевірки знань

Критерій порівняння бібліотек для побудови інтерфейсів	Бібліотеки для побудови інтерфейсів		
	React	Vue.js	Preact
Рівень розвитку спільноти користувачів та технічної підтримки	+	+–	–
Ступінь гнучкості та можливості конфігурації	+	+–	+–
Придатність до масштабування у великих програмних проєктах	+	+	+–
Здатність до впровадження в наявні програмні рішення	+	+–	+
Наявність повної документації та навчальних матеріалів	+	+	–

Таким чином, вибір React для побудови інтерфейсу ПЗ автоматизації перевірки знань є логічним та виправданим. Він гарантує створення зручного, сучасного та адаптивного інтерфейсу, що задовольняє вимоги користувачів та дозволяє безперешкодно розширювати функціонал програмного забезпечення у майбутньому. Комбінація високої продуктивності, гнучкості та активної підтримки спільноти робить React ефективним інструментом для реалізації багатофункціональних веб-орієнтованих інформаційних систем.

2.3 Висновки за розділом 2

Під час розробки та впровадження програмного забезпечення для автоматизації перевірки знань, основу технічної реалізації склала мова програмування JavaScript. Такий вибір виправданий завдяки її розгалуженому функціоналу, що є незамінним для створення інтерактивних веб-рішень, а також завдяки можливості реалізувати логіку на стороні клієнта і сервера, використовуючи єдиний технологічний набір. Використання уніфікованого програмного стеку допомагає підтримувати архітектурну єдність системи, спрощує інтеграцію окремих частин та мінімізує труднощі, пов’язані з подальшим обслуговуванням та зміною програмного коду.

У питанні формування візуальної частини (інтерфейсу) цього програмного продукту було задіяно бібліотеку React. Цей інструмент специфічно націлений на створення інтерфейсів, що відзначаються високою ергономічністю та інтуїтивною простотою для користувачів системи. Завдяки React досягається гнучкість у створенні елементів інтерфейсу, покращується швидкість реакції вебдодатку, і закладається надійна база для розширення функціоналу системи у майбутньому.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення

Для реалізації програмного забезпечення для автоматизації перевірки знань було обрано варіант реалізації у вигляді односторінкового додатку - SPA (Single Page Application).

Односторінкові програми здатні імітувати роботу настільних застосунків. Їхня архітектурна будова передбачає, що при навігації між різними розділами оновлюється лише частина контенту. Як наслідок, відпадає потреба у повторному завантаженні тих самих компонентів. Це приносить значні зручності як розробникам, так і кінцевим користувачам [13].

Ключові переваги SPA:

− простота розробки. Для створення SPA існують готові бібліотеки та фреймворки, що дає змогу вести розробку фронтенду та бекенду паралельно. Більше того, на базі створеного коду згодом можна створити й мобільний додаток;

− висока гнучкість інтерфейсу. Розробляти привабливий та інтерактивний дизайн значно легше для єдиної сторінки;

− ефективне кешування інформації. Критичні дані завантажуються за один раз, а далі користувач SPA може частково працювати без доступу до мережі, підключаючись до Інтернету лише для збереження результатів роботи.

Незважаючи на усі позитивні аспекти, SPA має певні мінуси, які стримують його стрімке поширення:

− складність SEO-просування. Адреси (URL) сторінок майже не змінюються, а дані завантажуються динамічно, тоді як для ефективної оптимізації важлива стабільність та унікальність URL для кожної сутності. Пошуковим роботам складно якісно сканувати SPA, тому коректно індексувати такі ресурси зазвичай вдається лише Google;

− значне навантаження на браузер. Ця проблема виникає через використання відносно “важких” клієнтських фреймворків;

– потенційно висока вартість. Хоча кінцева ціна розробки різна, для складних проєктів вона може бути досить суттєвою.

Веб-ресурс складається зі статичних візуальних елементів та динамічного компоненту App.jsx, який слугує точкою входу та відповідає за оброблення та відображення даних. У результаті цих операцій користувач отримує інформацію у зручному графічному форматі.

Веб-сайт структуровано навколо п’яти ключових сторінкових компонентів:

– головна сторінка, що надає функціонал для пошуку користувачів;

– сторінка профілю користувача, де відображається список усіх пройдених тестів;

– сторінка керування налаштуваннями особистого профілю;

– сторінка для створення або редагування тестів (вікторин);

– сторінка безпосереднього проходження тесту.

Для забезпечення роботи додатку було розроблено API, розташоване на спеціально виділеному віддаленому сервері. Інтерфейс програмування застосунків (Application Programming Interface, API) являє собою набір методів, які розробник може використовувати для доступу до сховища даних системи.

Загальна схема функціонування програми представлена на рисунку 3.1.

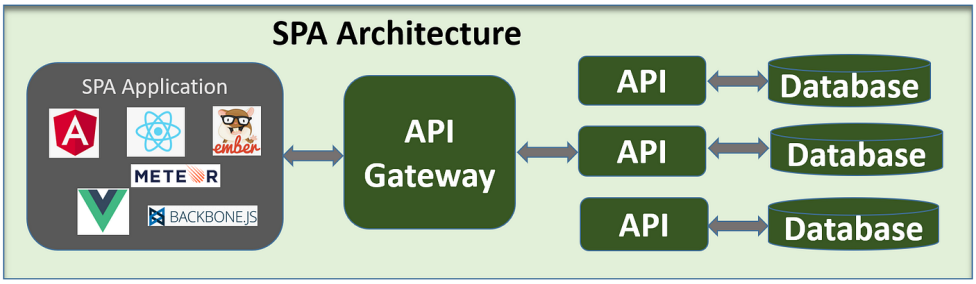

Рисунок 3.1 – Загальна структурна схема роботи програми [13]

Під час розробки програмного забезпечення, призначеного для автоматизації тестування знань, було застосовано технологію REST

(Representational State Transfer). Це архітектурна концепція мережевих протоколів, яка уможливлює доступ до різноманітних інформаційних ресурсів. REST базується на фундаментальних засадах функціонування Всесвітньої павутини, зокрема, на можливостях, які надає протокол HTTP. По суті, REST являє собою доволі лаконічний спосіб роботи з даними, що не вимагає додаткових внутрішніх прошарків (абстракцій). Кожен окремий елемент даних чітко ідентифікується за допомогою універсального ідентифікатора, наприклад, URL [14]. При цьому кожен URL мусить відповідати строго визначеному формату (рисунок 3.2).

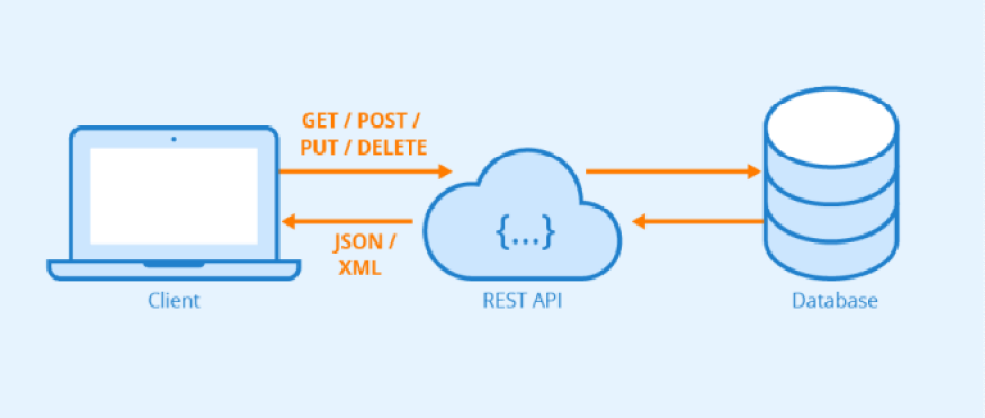

Рисунок 3.2 - Принцип роботи REST API [14]

Взаємодія між клієнтами та серверами відбувається шляхом надсилання порцій даних, що називаються повідомленнями. Ті повідомлення, що ініціюються клієнтом, позначаються як "запити", тоді як ті, що надходять від сервера, іменуються "відповідями".

В застосунку передбачено чотири різновиди запитів:

– GET застосовується з метою отримання даних за певним запитом. Опрацювавши запит, GET повертає запитуваний ресурс, як правило, у форматі XML або JSON;

– POST використовується у ситуаціях, коли необхідно створити новий елемент;

– PATCH служить для часткової модифікації існуючого об'єкта;

– PUT призначений для повної заміни даних в уже наявному об'єкті. Його тіло має містити актуальні дані, отримані з першоджерела, чи то повний набір, чи лише певні частини.

– DELETE використовується для вилучення об'єкта, що існує.

Для здійснення комунікації із сервером було прийнято рішення використовувати бібліотеку Axios.

Axios є HTTP-клієнтом, що базується на промісах (Promises) і призначений для середовищ Node.js та веб-браузерів. Ця бібліотека є ізоморфною (компілюється та працює ідентично як у браузері, так і в Node.js, використовуючи єдиний код). На серверній стороні бібліотека задіює вбудований HTTP-модуль Node.js, а на клієнті (у браузері) покладається на механізм XMLHttpRequests [15].

Вибір HTTP-клієнтом припав саме на Axios, оскільки стандартний функціонал «fetch()» вважається менш гнучким, до того ж, отримання даних через нього вимагає двох кроків (рисунок 3.3):

− спочатку дані отримуються у вигляді JSON-послідовності;

− потім виконується конвертація цього JSON у нативний об'єкт.

```
fetch(`http://localhost:3001/quizzes/${owner}`)
  .then((res) => res.json())
  .then((data) => console.log(data));
```

Рисунок 3.3 – Отримання даних за допомогою методу fetch()

Отже, завдяки Axios немає потреби додатково викликати метод «json()» для обробки результатів HTTP-запиту. Axios надає об'єкт із потрібними даними (рисунок 3.4).

```
const { data } = await api.get(`/quizzes/${owner}`);
```

Рисунок 3.4 – Отримання даних за допомогою бібліотеки axios

Для керування навігацією в проєкті задіяна бібліотека «react-router». Вона досить відома, проста в освоєнні та має якісний довідковий матеріал. Серед функціональних можливостей, які вона пропонує, варто відзначити навігацію за натисканням (через компонент `Link`), переадресацію (завдяки компоненту `Redirect`), визначення шляхів (компонент `Route`) та доступ до історії переглядів (через властивість `history`). Схема маршрутизації в додатку для автоматизації перевірки знань зображено на рисунку 3.5.

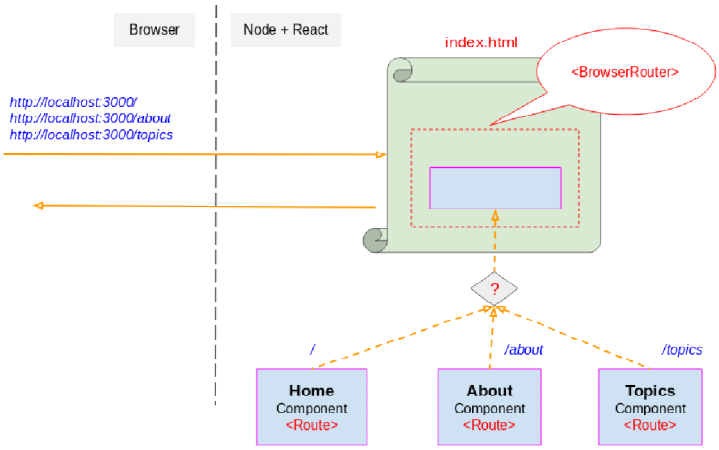

Рисунок 3.5 – Схема маршрутизації в додатку для автоматизації перевірки знань

3.2 Розробка Бази даних

Під час створення вебдодатку було задіяно прикладний програмний інтерфейс (API), розміщений на віддаленому сервері, який забезпечує доступ до даних на основі документо-орієнтованої моделі зберігання. Такий підхід дозволяє організувати обмін інформацією між клієнтською та серверною частинами застосунку з використанням гнучкої структури даних.

Бази даних типу NoSQL розроблялися з урахуванням специфічних моделей даних і характеризуються відсутністю жорстко фіксованих схем. Це надає можливість адаптувати структуру зберігання інформації відповідно до потреб сучасних програмних рішень. Широке поширення NoSQL-баз зумовлене спрощеним процесом розробки, високою функціональністю та

стабільною продуктивністю незалежно від обсягів даних. На відміну від класичних реляційних систем керування базами даних, такі сховища підтримують низку додаткових можливостей, зокрема зберігання даних у форматі пар «ключ–значення» для короткочасного кешування, а також роботу з неструктурованими наборами інформації [16].

Особливу популярність серед NoSQL-рішень отримали документо-орієнтовані бази даних. За принципом організації вони мають певну схожість зі стовпцевими сховищами, однак забезпечують значно вищу гнучкість у формуванні структури даних. Використання документів дозволяє усунути обмеження, притаманні іншим типам сховищ, і створювати записи довільної структури та будь-якого рівня складності. Це робить такі бази даних зручними для роботи з різнорідною та динамічною інформацією [16].

Для отримання та обробки даних через API у межах розроблюваного вебдодатку було визначено чотири основні типи документів: «User», «Quiz», «Question» та «Answer». Кожен із них відповідає за зберігання окремої логічної частини інформації та використовується для забезпечення коректної роботи функціональних модулів системи.

Документ «User» містить основні відомості про користувача, зокрема унікальний ідентифікатор, адресу електронної пошти, облікові дані для автентифікації, ім’я користувача та посилання на аватар. Детальний опис структури документа «User» наведено в таблиці 3.1. Схема документа «User» представлена на рисунку 3.6.

Таблиця 3.1 – Опис документа «User»

Назва ключа	Тип значення	Опис
_id	String	Забезпечує унікальність документа
email	String	Електронна пошта користувача
password	String	Пароль користувача
username	String	Ім’я користувача
avatar	String	Шлях до фото користувача

```
const User = new Schema(
  {
    email: { type: String, unique: true, required: true },
    username: { type: String, unique: true, required: true },
    password: { type: String, required: true },
    image: { type: String },
  },
  { timestamps: true }
);
```

Рисунок 3.6 – Схема документа «User»

Документ «Quiz» (_id, name, owner, questions, image) зберігає інформацію про тест. Опис документа «Quiz» представлений в таблиці 3.2. Схема документа «Quiz» представлена на рисунку 3.7.

Таблиця 3.2 – Опис документа «Quiz»

Назва ключа	Тип значення	Опис
_id	String	Забезпечує унікальність документа
name	String	Назва вікторини
owner	String	Ім’я власника вікторини
questions	Array	Масив ідентифікаторів питань вікторини
image	String	Шлях до фото вікторини

```
const Quiz = new Schema(
  {
    name: { type: String, default: "New quiz" },
    owner: { type: String, required: true },
    questions: [{ type: Schema.Types.ObjectId, ref: "Question" }],
    image: { type: String },
  },
  { timestamps: true }
);
```

Рисунок 3.7 – Схема документа «Quiz»

Документ «Question» (_id, quizId, answers, text) зберігає інформацію про питання вікторини. Опис документа «Question» представлений в таблиці 3.3. Схема документа «Question» представлена на рисунку 3.8.

Таблиця 3.3 – Опис документа «Question»

Назва ключа	Тип значення	Опис
id	String	Забезпечує унікальність документа
quizId	String	Ідентифікатор вікторини, до якої належить питання
answers	Array	Масив ідентифікаторів відповідей на питання вікторини
text	String	Текст питання

```
const Question = new Schema(
  {
    quizId: { type: Schema.Types.ObjectId, ref: "Quiz" },
    answers: [{ type: Schema.Types.ObjectId, ref: "Answer" }],
    text: { type: String, default: "New question" },
    isAnswered: { type: Boolean, default: false },
  },
  { timestamps: true }
);
```

Рисунок 3.8 – Схема документа «Question»

Документ «Answer» (_id, questionId, text, isRight) зберігає інформацію про відповідь на питання вікторини. Опис документа «Answer» представлений в таблиці 3.4. Схема документа «Answer» представлена на рисунку 3.9.

Таблиця 3.4 – Опис документа «Answer»

Назва ключа	Тип значення	Опис
id	String	Забезпечує унікальність документа
text	String	Текст відповіді
questionId	String	Ідентифікатор питання, до якого належить відповідь
isRight	Boolean	Показує, чи є відповідь правильною

```
const Answer = new Schema(
  {
    questionId: { type: Schema.Types.ObjectId, ref: "Question" },
    text: { type: String, default: "New answer" },
    isRight: { type: Boolean, default: false },
  },
  { timestamps: true }
);
```

Рисунок 3.9 – Схема документа «Answer»

3.3 Основний алгоритм роботи програмного забезпечення

Основний алгоритм роботи програмного забезпечення зображений на рисунку 3.10. Система реалізована за клієнт-серверною архітектурою та поєднує сучасні підходи до розробки програмного забезпечення, зокрема компонентний підхід, централізоване керування станом і асинхронну взаємодію з сервером.

Алгоритм охоплює такі основні етапи: ініціалізація інтерфейсу користувача, введення та перевірка даних, обробка подій, передача інформації на сервер, серверна обробка та оновлення глобального стану застосунку. Кожен з етапів є логічно завершеним і взаємопов’язаним з іншими частинами системи.

Після успішної авторизації користувача відбувається завантаження компонента QuizEditor, який відповідає за створення та редагування тестів. На цьому етапі виконується підключення до глобального стану Redux з використанням селекторів. Зі сховища отримуються дані про поточного користувача та, за наявності, інформація про вибраний тест.

Далі ініціалізується форма введення даних за допомогою бібліотеки react-hook-form. Початкові значення форми встановлюються динамічно: якщо користувач редагує існуючий тест, поля заповнюються збереженими даними; якщо створюється новий тест — використовуються значення за замовчуванням. Такий підхід забезпечує універсальність алгоритму та повторне використання коду.

Одночасно підключається схема валідації, описана за допомогою бібліотеки Yup. Вона визначає обов’язкові поля, типи даних та логічні обмеження, що дозволяє відсіяти некоректні дані ще на клієнтській стороні.

На наступному етапі алгоритму користувач взаємодіє з інтерфейсом. Він вводить назву тесту, додає питання, формує варіанти відповідей та може завантажувати зображення для візуального супроводу. Кожен логічний блок реалізований у вигляді окремого React-компонента, що підвищує модульність і спрощує підтримку системи.

Усі дії користувача негайно фіксуються у внутрішньому стані форми. React Hook Form оптимізує процес обробки введення, мінімізуючи кількість перерендерень та підвищуючи продуктивність інтерфейсу. Це особливо важливо при роботі з великими наборами питань.

Після натискання кнопки підтвердження запускається механізм перевірки введених даних. Алгоритм послідовно аналізує всі поля форми відповідно до заданих правил валідації. У разі виявлення помилок процес надсилання даних зупиняється, а користувачу відображаються відповідні повідомлення.

Такий підхід дозволяє значно зменшити кількість помилкових запитів до сервера та підвищити надійність системи. Лише після успішного проходження всіх перевірок алгоритм переходить до наступного етапу.

Після успішної валідації викликається функція обробки надсилання форми. Вона формує об’єкт тесту, який містить усі необхідні дані: назву, перелік питань, інформацію про автора та вкладені файли. Цей об’єкт передається до Redux-логіки, де створюються відповідні action-об’єкти.

Для взаємодії з сервером використовується Axios-клієнт. Алгоритм передбачає автоматичне додавання токена автентифікації до кожного HTTP-запиту, що забезпечує захищений доступ до API. У разі відсутності або некоректності токена ініціюється процедура виходу користувача із системи.

На серверній стороні отримані дані обробляються відповідними сервісами. Алгоритм передбачає виконання перевірок доступу, обробку бізнес-логіки та збереження інформації у базі даних. Для користувачів реалізовано функції реєстрації, оновлення профілю, пошуку та отримання публічних даних.

Особливу увагу приділено безпеці: паролі користувачів хешуються з використанням криптографічних алгоритмів, а доступ до ресурсів контролюється за допомогою токенів. У разі помилки сервер повертає відповідний статус та повідомлення.

Після отримання відповіді від сервера алгоритм переходить до етапу оновлення глобального стану застосунку. Redux-ред’юсери аналізують тип отриманої дії та змінюють відповідні частини стану. Оновлюється інформація про користувача, результати тестування або список доступних тестів (рис. 3.10).

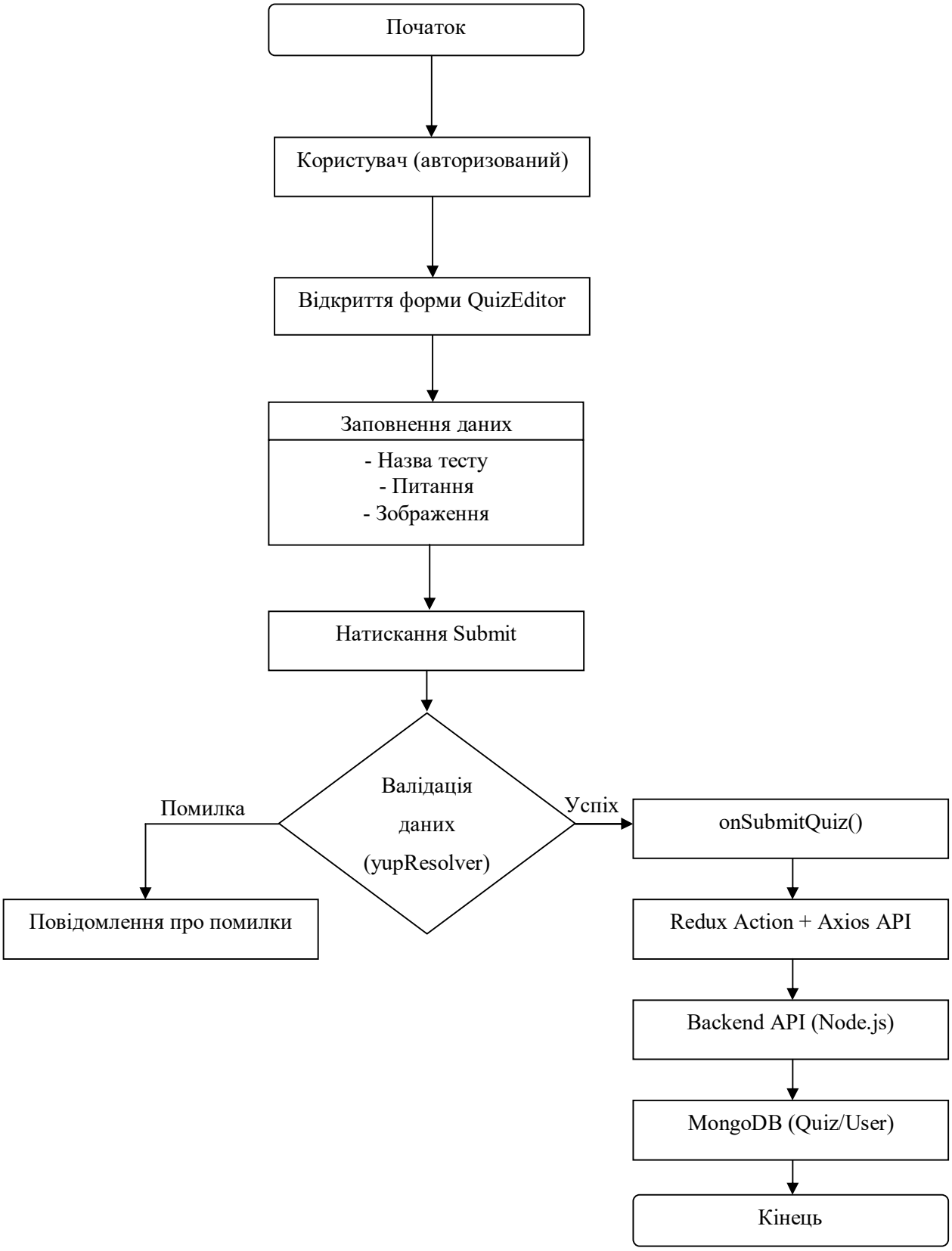

Рисунок 3.10 – Основний алгоритм роботи програмного забезпечення

Централізоване керування станом забезпечує узгодженість даних між усіма компонентами інтерфейсу та спрощує масштабування системи.

На завершальному етапі користувач отримує зворотний зв’язок про результат виконаної операції. Це може бути успішне створення або оновлення тесту, зміна даних профілю чи повідомлення про помилку. Алгоритм завершується поверненням системи у стабільний стан, готовий до подальшої взаємодії з користувачем.

Отже, описаний алгоритм забезпечує повний та надійний цикл роботи веб-застосунку, поєднуючи зручність користування, безпеку даних і сучасні підходи до програмної інженерії.

3.4 Проєктування інтерфейсу

На сучасному етапі розвитку програмного забезпечення значна увага зосереджується на проєктуванні інтерфейсу взаємодії з користувачем. Адже саме від якості цього інтерфейсу критично залежить, наскільки конкурентоспроможним буде кінцевий продукт і як його приймуть ті, хто ним користуватиметься. Інтерфейс має бути інтуїтивно зрозумілим, простим у використанні, мати гармонійне поєднання кольорів, а також надавати користувачеві вичерпні відомості про те, що відбувається з керованими системами в даний момент. При цьому необхідно уникати ситуації, коли екран перенасичений надмірною кількістю візуальних компонентів. Вагоме значення також мають естетичні аспекти оформлення та послідовне виконання типових вимог та норм стосовно конструювання інтерфейсів [17].

Інтерфейс, реалізований у розробленому програмному рішенні, був збудований із застосуванням бібліотеки React, багатопарадигмальної мови JavaScript, мови розмітки гіпертексту HTML5, а також каскадних таблиць стилів CSS3.

Архітектурно інтерфейс поділяється на такі ключові розділи:

− стартова (головна) сторінка, що надає функціонал для пошуку серед користувачів;

– сторінка індивідуального користувача, де представлений перелік тестів (вікторин), до яких надано доступ;

– сторінка для керування параметрами облікового запису користувача;

– розділ, призначений для формування та зміни тестів (вікторин);

– сторінка, безпосередньо призначена для проходження вибраного тесту.

На рисунку 3.11 наведено концепт сторінки головної сторінки веб-сайту.

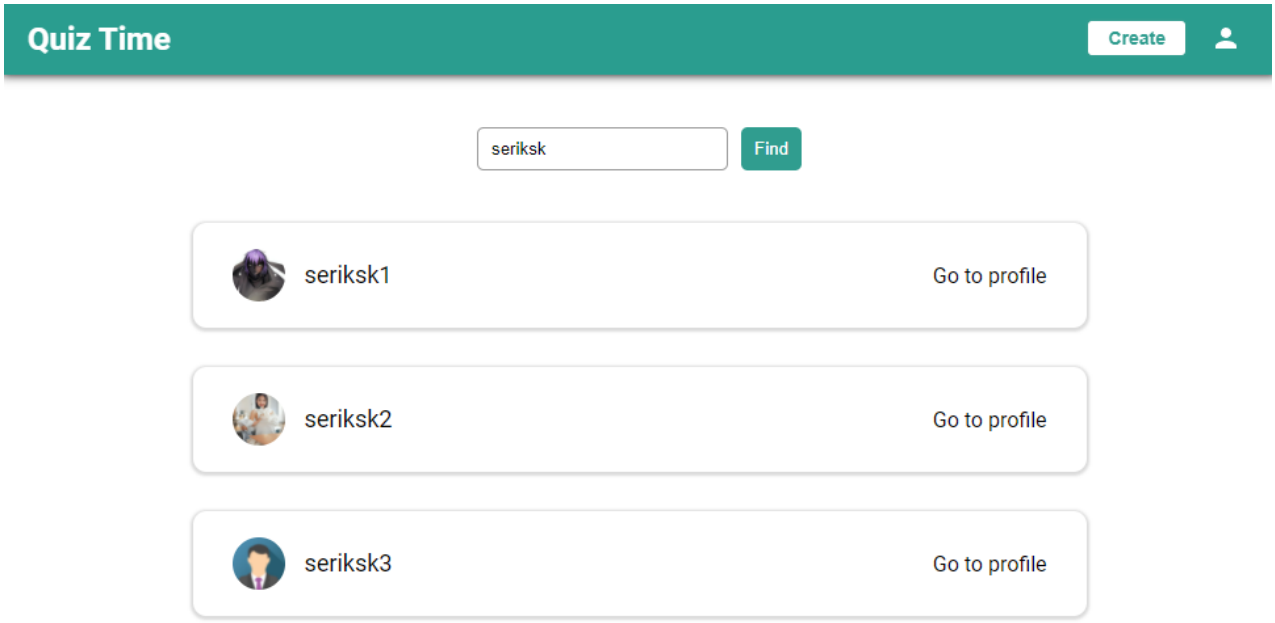

Рисунок 3.11 – Концепт головної сторінки веб-сайту

3.5 Висновки за розділом 3

У даному розділі розглянуто архітектуру та засади створення програмного забезпечення для автоматизації перевірки знань. Обґрунтовано вибір односторінкового вебзастосунку (SPA), який гарантує високу швидкодію, зручність використання та дієву комунікацію з інтерфейсом завдяки динамічному оновленню вмісту без перезавантаження сторінок.

Проаналізовано структуру застосунку та визначено головні функціональні складові, що впроваджують ключові можливості системи.

Описано організацію клієнт-серверної взаємодії із застосуванням REST API та бібліотеки Axios, що забезпечує стандартизований і безпечний обмін даними між частинами системи.

Розглянуто нюанси проєктування бази даних на базі документо-орієнтованої NoSQL-моделі. Визначено головні види документів та їхнє призначення, що дає змогу гнучко зберігати та опрацьовувати відомості про користувачів, тести, запитання та відповіді.

Також описано основний алгоритм роботи програмного забезпечення, який охоплює всі кроки взаємодії користувача із системою та забезпечує узгодженість даних, надійність і потенціал масштабування застосунку. Отримані висновки підтверджують доцільність обраних архітектурних та програмних рішень.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Розроблений веб-сайт призначений для створення та проходження тестів, що дозволяє ефективно перевіряти знання користувачів. Зареєстровані користувачі можуть формувати власні тести, редагувати їх та проходити для оцінки результатів. Крім того, реалізовано функцію пошуку інших користувачів, що забезпечує можливість проходження тестів, створених іншими учасниками системи.

Усі дані для роботи веб-додатку надає API, розміщене на сервері, що дозволяє автоматично відображати оновлення інформації на сайті без додаткових дій з боку користувача. Розробка виконана у середовищі програмування Visual Studio Code із використанням сучасних вебтехнологій.

Сайт реалізує такі основні функції:

− реєстрація та автентифікація користувачів;

− завантаження та оновлення аватара користувача через налаштування профілю;

− створення та редагування тестів (вікторин);

− проходження тестів із відображенням результатів;

− пошук користувачів за іменами.

Додаток зберігається у віддаленому репозиторії. Для локальної роботи проект необхідно завантажити за допомогою команди «git clone <посилання на репозиторій>». Після цього через командний рядок слід виконати «npm i» для встановлення всіх залежностей, необхідних для функціонування системи. Для запуску програми використовується команда «npm run start».

Вхідними даними для веб-додатку є інформація, що зберігається на віддаленому сервері через API.

Вихідними даними є відомості про доступні тести, що відображаються на сторінці користувача та доступні для проходження.

4.2 Умови виконання програми

Для нормальної роботи веб-сайту та можливості його подальшого вдосконалення необхідно забезпечити мінімальні апаратні вимоги: персональний комп'ютер з процесором частотою не менше 2 ГГц, оперативною пам'яттю об'ємом щонайменше 4 ГБ та вільним дисковим простором не менше 5 ГБ, достатнім для встановлення браузера. Крім того, на комп'ютері має бути встановлено відповідне програмне забезпечення, зокрема операційна система та сучасний веб-браузер.

4.3 Виконання програмного забезпечення

Для запуску веб-додатку слід у адресному рядку браузера ввести адресу http://localhost:3000. Після цього відкривається сторінка авторизації (рисунок 4.1).

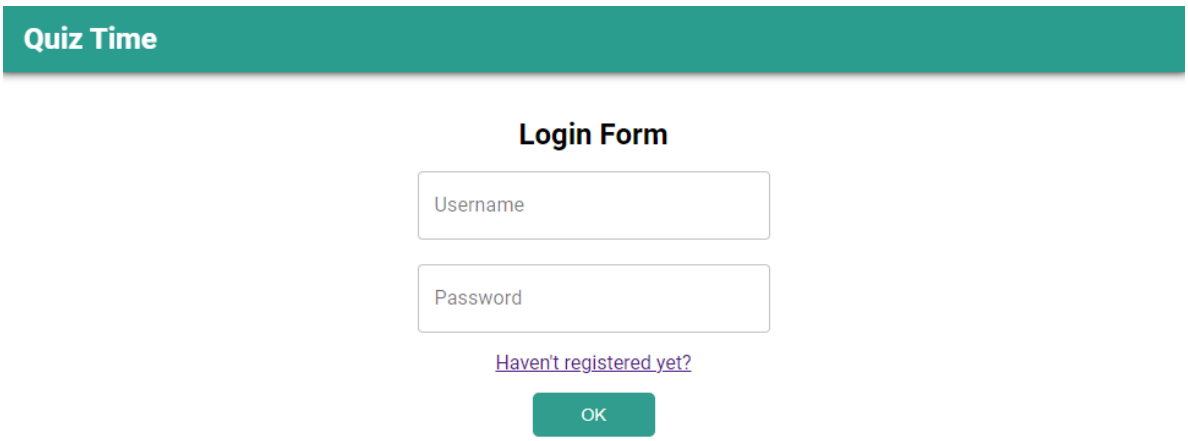

Рисунок 4.1 – Сторінка авторизації

Для входу необхідно вказати ім'я користувача та пароль. Якщо користувач ще не зареєстрований, потрібно перейти за посиланням "Haven't registered yet?", що перенаправляє на сторінку реєстрації (рисунок 4.2).

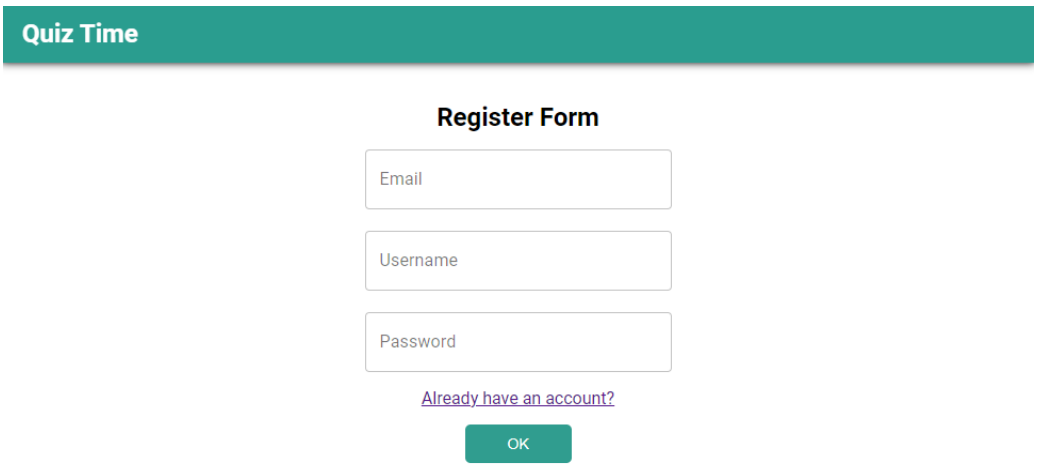

Рисунок 4.2 – Сторінка реєстрації

На сторінці реєстрації слід заповнити електронну пошту, ім’я користувача та пароль. Після успішної реєстрації користувач потрапляє на головну сторінку, де доступна функція пошуку інших учасників системи (рисунок 4.3).

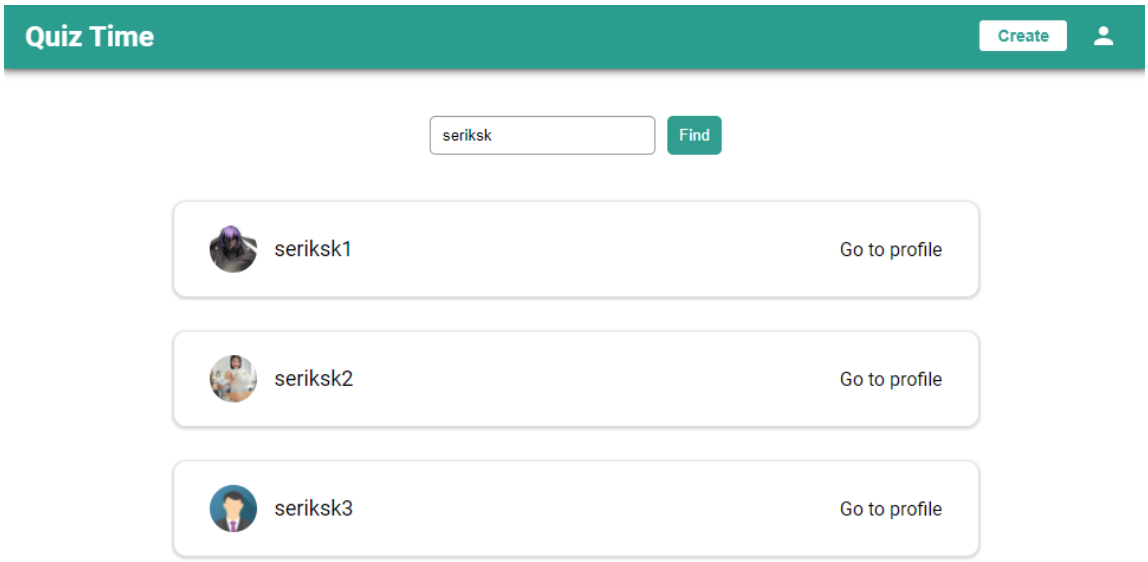

Рисунок 4.3 – Головна сторінка

Для переходу до профілю конкретного користувача необхідно натиснути на посилання “Go to profile” (рисунок 4.4). На сторінці профілю відображаються всі тести, створені користувачем. Для запуску обраного тесту

потрібно натиснути на три крапки під тестос, після чого відкривається панель із кнопкою “Play” (рисунок 4.5).

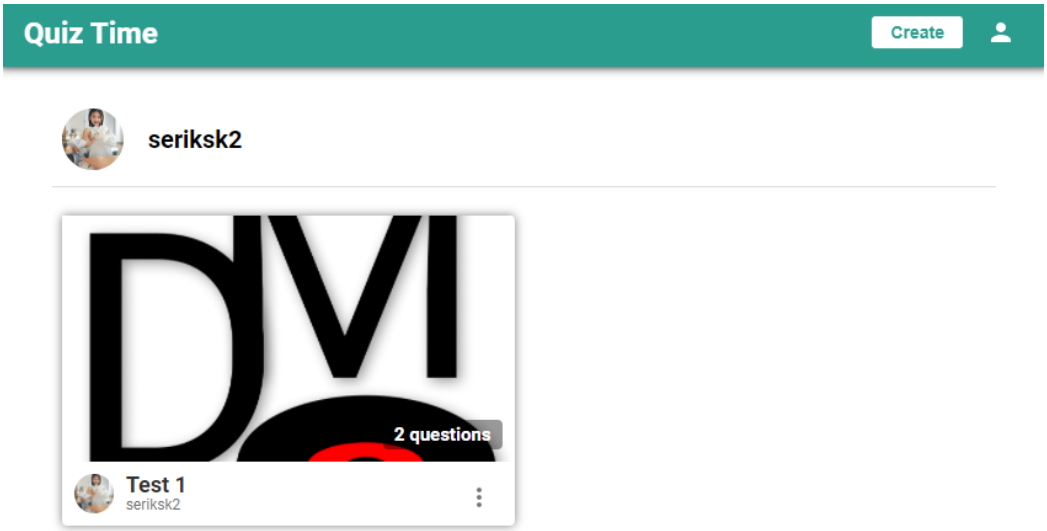

Рисунок 4.4 – Сторінка користувача

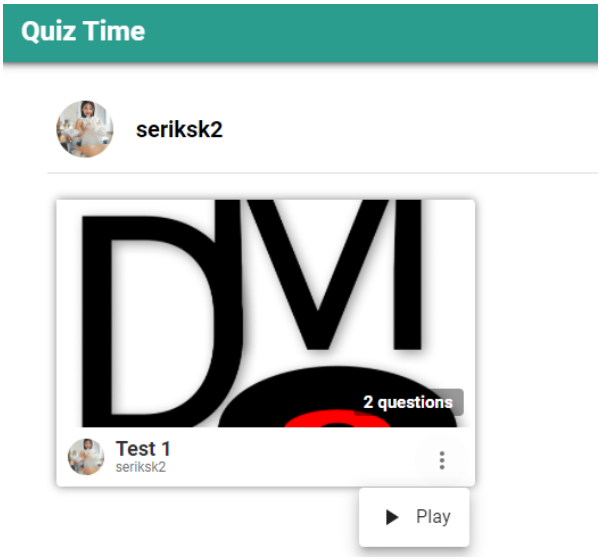

Рисунок 4.5 – Меню тесту

На сторінці тесту користувач бачить перше питання та доступні варіанти відповідей (рисунок 4.6). Після цього запускається 10-секундний таймер, протягом якого необхідно вибрати відповідь. Потім відбувається перехід до наступного питання (рисунок 4.7).

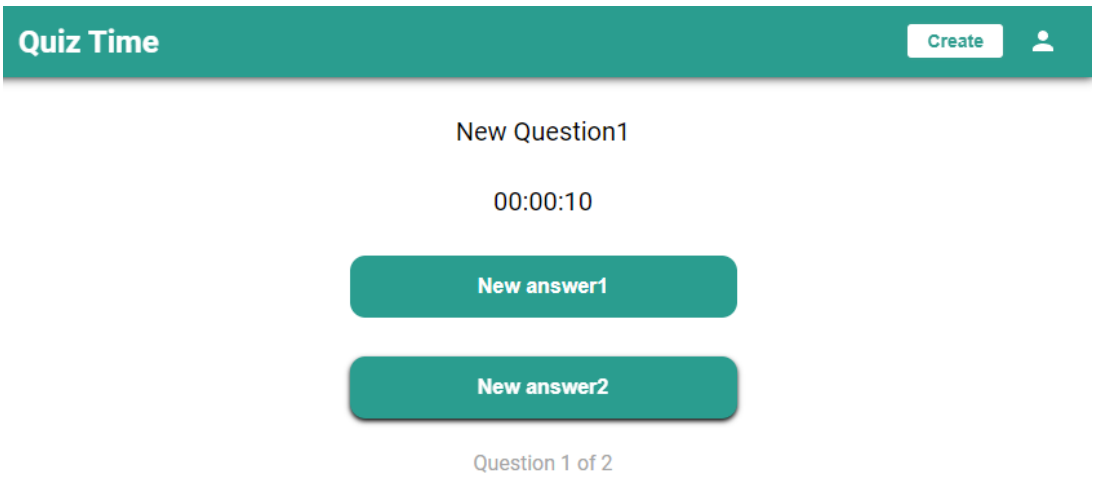

Рисунок 4.6 – Сторінка тесту

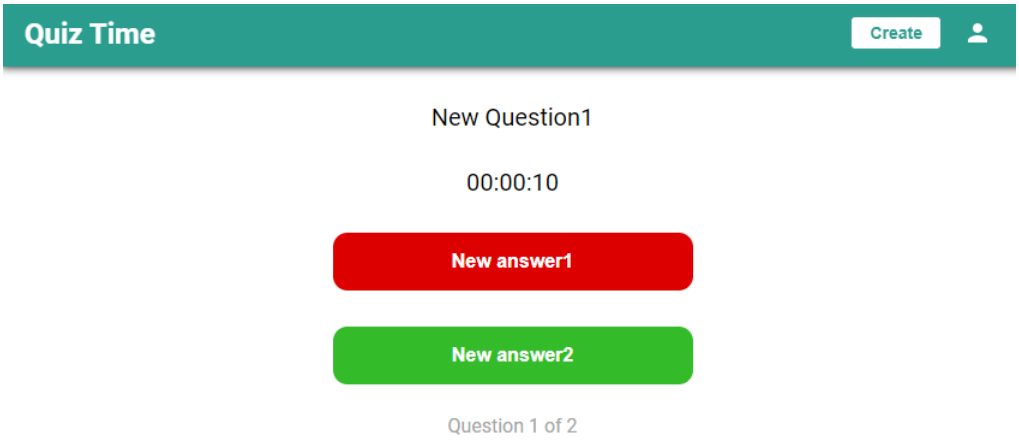

Рисунок 4.7 – Результат відповіді

Після завершення всіх питань користувач отримує підсумковий результат проходження тесту (рисунок 4.8).

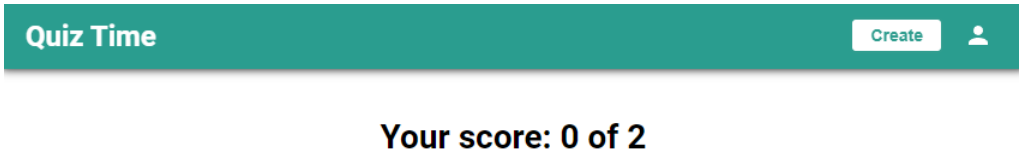

Рисунок 4.8 – Результат вікторини

Для створення власного тесту потрібно натиснути кнопку “Create” на верхній панелі сторінки (рисунок 4.9), після чого відкривається сторінка створення вікторини (рисунок 4.10).

Рисунок 4.9 – Кнопка “Create”

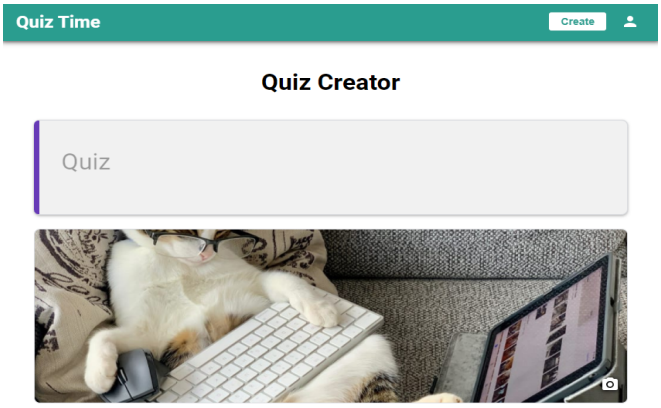

Рисунок 4.10 – Сторінка створення вікторини

Назву тесту можна ввести, натиснувши на панель з написом “Quiz” (рисунок 4.11). Для додавання зображення необхідно натиснути на іконку фотоапарата та обрати потрібне фото (рисунок 4.12).

Рисунок 4.11 – Введення назви тесту

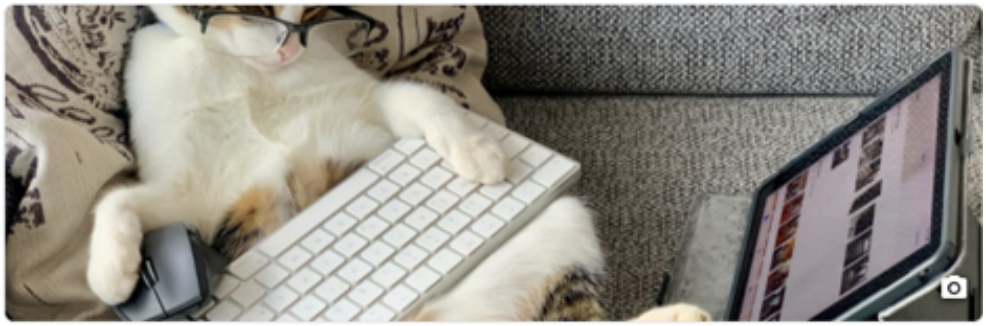

Рисунок 4.12 – Панель фото тесту

Щоб створити нове питання, слід натиснути кнопку “Add question”, після чого з’являється панель редагування, де можна ввести текст питання та варіанти відповідей (рисунок 4.13).

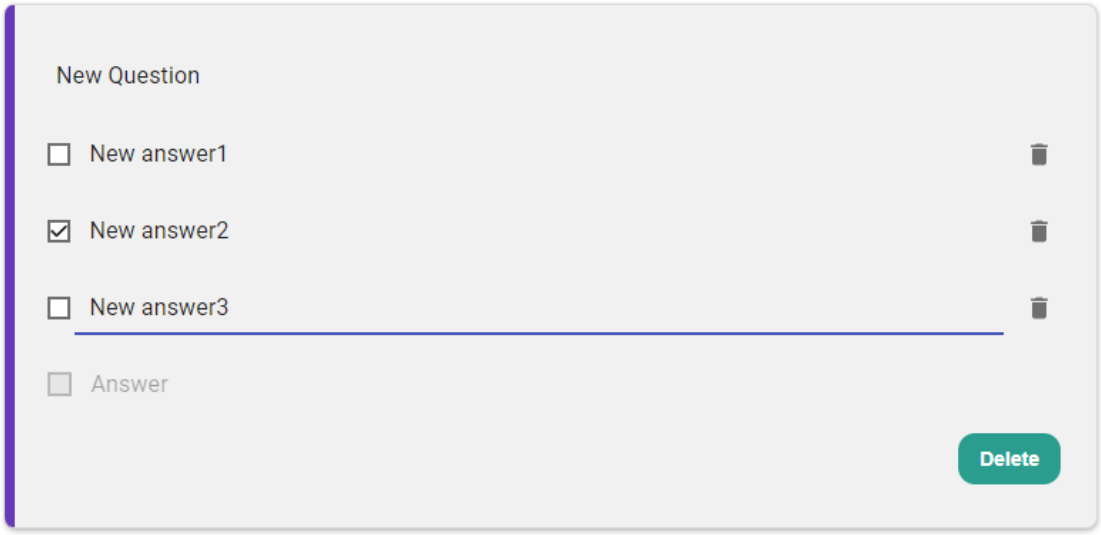

Рисунок 4.13 – Панель редагування питання

Після заповнення тесту слід натиснути кнопку “Create quiz” для збереження, і тест відобразиться на сторінці користувача (рисунок 4.14).

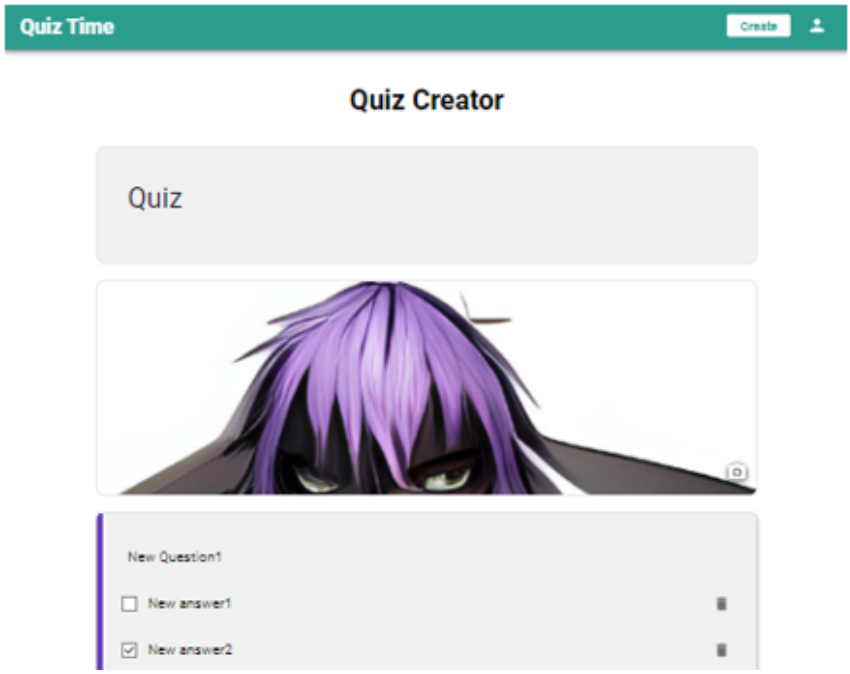

Рисунок 4.14 – Заповнена вікторина

Створені тести можна редагувати, натиснувши на три крапки під тестом і обравши кнопку “Edit” у меню (рисунок 4.15).

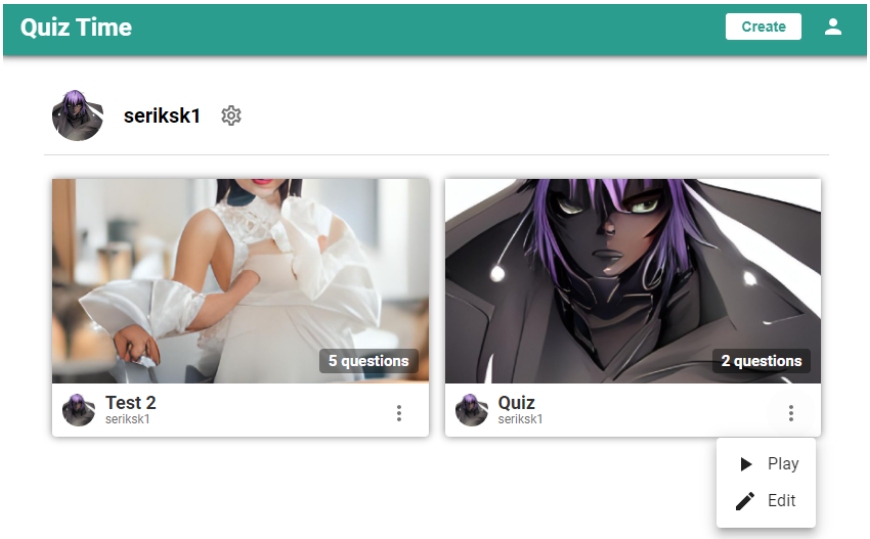

Рисунок 4.15 – Новий тест на сторінці користувача

Для зміни фото профілю потрібно відкрити верхнє меню, натиснувши на іконку користувача, і вибрати пункт “Settings” (рисунок 4.16). Далі слід натиснути кнопку “Upload”, обрати файл зображення та зберегти зміни кнопкою “Save” (рисунок 4.17).

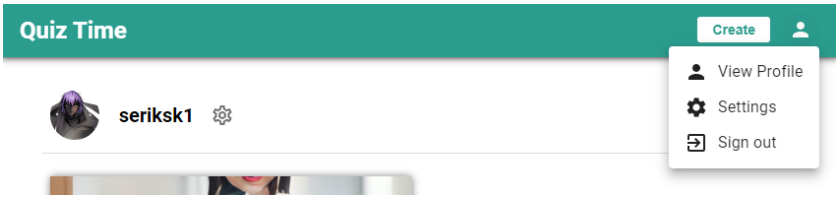

Рисунок 4.16 – Меню на верхній панелі

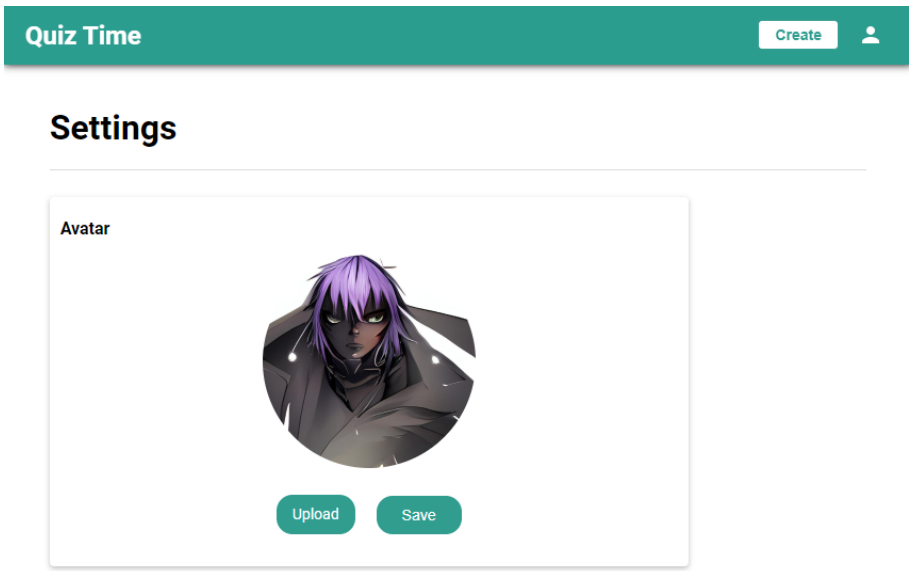

Рисунок 4.17 – Сторінка налаштувань

4.4 Тестування програмного забезпечення

Було проведено комплексне тестування створеного програмного забезпечення для автоматизації оцінювання знань. Система пройшла як модульне, так і інтеграційне тестування із використанням стандартних інструментів React та бібліотеки React Testing Library. Особлива увага приділялася перевірці окремих компонентів, що відповідають за основний функціонал додатку [18].

За результатами тестування підтверджено стабільну роботу програми без критичних збоїв. Додаток повністю відповідає заявленим функціональним вимогам та коректно виконує всі передбачені завдання. Виявлені незначні помилки були оперативно виправлені в процесі доопрацювання коду.

Особливу увагу було приділено тестуванню сценаріїв взаємодії користувача з інтерфейсом, зокрема створення, редагування та перегляду тестів. Перевірялося правильне оброблення введених даних, дотримання правил валідації форм, а також реакція програми на стандартні помилки користувачів [18].

Таким чином, розроблене програмне забезпечення характеризується високою надійністю, повною функціональністю та відповідністю сучасним стандартам якості веб-застосунків.

4.5 Висновки за розділом 4

У четвертому розділі проаналізовано призначення, умови застосування та функціональні можливості програмного забезпечення для автоматизації процесу перевірки знань. Було описано порядок виконання та умови роботи розробленого програмного забезпечення.

Виконано запуск та тестування додатку, включно з модульним і інтеграційним тестуванням за допомогою React Testing Library. Перевірено основні компоненти на правильність виконання функцій. Результати показали, що програма працює стабільно, відповідає заявленим функціональним вимогам і готова до практичного використання.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було виконано програмну реалізацію застосунку для автоматизації перевірки знань.

Для цього було виконано ряд завдань:

- досліджено актуальні способи організації перевірки знань у розрізі діджиталізації освітньої сфери;
- проведено аналіз класичних і дистанційних технік оцінювання;
- виконано аналіз існуючих платформ для автоматизації перевірки знань та виявлено типові недоліки;
- виконано детальний опис структури застосунку, його головних робочих модулів та шляхів їхньої взаємодії;
- спроєктовано базу даних на основі документоорієнтованої NoSQL-моделі;
- сформовано алгоритм функціонування системи;
- виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням;
- розроблено програмне забезпечення для автоматизації перевірки знань;
- описано процедуру запуску системи, реєстрації користувачів, створення та коригування тестів, а також проходження опитувань із візуалізацією отриманих результатів;
- проведено всебічне тестування застосунку для автоматизації перевірки знань, включаючи покомпонентне та інтеграційне тестування із застосуванням React Testing Library;
- перевірено коректність функціонування ключових елементів системи та послідовностей дій користувача з інтерфейсом.

Розробка програмного забезпечення відбувалася із застосуванням мови програмування JavaScript, що дало змогу реалізувати логіку як на боці користувача, так і на сервері, забезпечуючи уніфікований технологічний

фундамент. Для формування інтерфейсу було використано бібліотеку React, яка гарантує високу динамічність, зручність для кінцевого користувача та гнучкість у плані подальшого вдосконалення функціоналу. Обраний архітектурний підхід односторінкового вебзастосунку (SPA) забезпечує швидке завантаження та динамічне оновлення контенту без потреби повного перезавантаження сторінки, що підвищує загальну якість взаємодії користувача із системою.

Отримані результати тестування розроблюваного додатку підтвердили стабільність роботи програми, її відповідність заявленим функціональним вимогам та готовність до застосування на практиці.

Застосунок реалізує такі функціональні вимоги:

− реєстрація та автентифікація користувачів;

− завантаження та оновлення аватара користувача через налаштування профілю;

− створення та редагування тестів;

− проходження тестів із відображенням результатів;

− пошук користувачів за іменами.

В результаті розробки реалізовано систему, що відповідає всім вимогам визначеним на початку розробки у вигляді функціональних вимог.

Усі завдання дипломної кваліфікаційної роботи бакалавра повністю виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What is Remote Testing? Definition & Methodology. *Userback*. URL: https://userback.io/glossary/usability-testing/remote-testing/ (date of access: 06.05.2026).

2. What is Remote Usability Testing? *Trymata formerly TryMyUI.* URL: https://trymata.com/blog/what-is-remote-usability-testing/ (date of access: 09.05.2026).

3. Understanding remote usability testing and how to conduct it. *Maze.* URL: https://maze.co/guides/usability-testing/remote/ (date of access: 09.05.2026).

4. Online testing vs. In-person testing: which is best? *Prov.* URL: https://provexam.com/online-testing-vs-in-person-testing-which-is-best/ (date of access: 11.05.2026).

5. What is software testing, definition, types, methods, and approaches? *Quora*. URL: https://www.quora.com/What-is-software-testing-definition-types-methods-and-approaches (date of access: 11.05.2026).

6. Overview of 15 Best Test Maker Software. *Speedexam*. URL: https://www.speedexam.net/blog/15-best-test-maker-software-in-2025/ (date of access: 14.05.2026).

7. What Is JavaScript Used For? *Coursera*. URL: https://www.coursera.org/articles/what-is-javascript-used-for (date of access: 15.05.2026).

8. C# Programming: What It Is, How It's Used + How to Learn *Coursera*. URL: https://www.coursera.org/articles/c-sharp (date of access: 15.05.2026).

9. Comparing Python to Other Languages. *Python*. URL: https://www.python.org/doc/essays/comparisons/ (date of access: 16.05.2026).

10. React A JavaScript library for building user interfaces. *ReactJS*. URL: https://legacy.reactjs.org/ (date of access: 16.05.2026).

11. Vue.js is a versatile and efficient JavaScript framework for building web user interfaces. *Sanity*. URL: https://www.sanity.io/glossary/vue-js (date of access: 17.05.2026).

12. What is Preact and when should you consider using it? *Merixstudio*. URL: https://www.merixstudio.com/blog/what-preact-and-when-should-you-consider-using-it (date of access: 17.05.2026).

13. Single Page Application - definition, examples, use cases. *Uniquedevs*. URL: https://uniquedevs.com/en/blog/single-page-application-definition-examples-use-cases/ (date of access: 17.05.2026).

14. REST API Tutorial. *REST API Tutorial*. URL: https://restfulapi.net/ (date of access: 26.05.2026).

15. Axios in REACT JavaScript *Medium.* URL: https://medium.com/@awofesobipeace/axios-in-react-js-f5a0cd821694 (date of access: 19.05.2026).

16. What Is NoSQL? NoSQL Databases Explained. *Mongodb*. URL: https://www.mongodb.com/resources/basics/databases/nosql-explained (date of access: 19.05.2026).

17. Принципи веб-дизайну інтерфейсу користувача. *Stfalcon*. URL: https://stfalcon.com/uk/blog/post/user-interface-web-design-principles (дата звернення: 19.05.2026).

18. Тестування веб-проєктів: основні етапи та поради. *QALight Центр підготовки IT-фахівців*. URL: https://qalight.ua/baza-znaniy/testuvannya-veb-proektiv-osnovni-etapi-ta-poradi/ (дата звернення: 19.05.2026).

ДОДАТОК А

Технічне завдання

Вступ

Даний застосунок присвячений автоматизації перевірки знань.

А.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Розробка програмного забезпечення для автоматизації перевірки знань», затверджене наказом Національного університету «Запорізька політехніка» № 139 від 07 квітня 2026 р.

А.2 Основні вимоги до програми, що розробляється

Розробка ПЗ для створення автоматизації перевірки знань є актуальною й перспективною темою, яка поєднує в собі технічні виклики та суспільну значущість. Такий інструмент не лише спрощує організацію процесів, але й створює умови для підвищення ефективності роботи, зменшення рутинного навантаження, покращення комунікації та прозорості в роботі.

Отже актуальною задачею залишається програмна реалізація застосунку для автоматизації перевірки знань.

А.2.1 Вимоги до функціональних характеристик

Вимоги до функціональних характеристик наступні:

− реєстрація та автентифікація користувачів;

− завантаження та оновлення аватара користувача через налаштування профілю;

− створення та редагування тестів (вікторин);

− проходження тестів із відображенням результатів;

− пошук користувачів за іменами.

А.2.2 Вимоги до інтерфейсу програми

Інтерфейс системи повинен відрізнятися лаконічністю, інтуїтивною зрозумілістю та зручністю для користувачів. Дизайн має бути адаптивним для зручності роботи на будь-якому дисплеї.

А.2.3 Вимоги до надійності

Забезпечення конфіденційності та безпеки даних забезпечується дотримання відповідних правил та рекомендацій щодо захисту даних користувачів та захисту інформації користувача.

А.2.4 Вимоги до складу та параметрів технічний засобів

Для нормальної роботи веб-сайту та можливості його подальшого вдосконалення необхідно забезпечити мінімальні апаратні вимоги: персональний комп’ютер з процесором частотою не менше 2 ГГц, оперативною пам’яттю об’ємом щонайменше 4 ГБ та вільним дисковим простором не менше 5 ГБ, достатнім для встановлення браузера. Крім того, на комп’ютері має бути встановлено відповідне програмне забезпечення, зокрема операційна система та сучасний веб-браузер.

ДОДАТОК Б

Опис програми

Б.1 Загальні відомості

Розробка ПЗ для створення автоматизації перевірки знань є актуальною й перспективною темою, яка поєднує в собі технічні виклики та суспільну значущість. Такий інструмент не лише спрощує організацію процесів, але й створює умови для підвищення ефективності роботи, зменшення рутинного навантаження, покращення комунікації та прозорості в роботі.

Отже актуальною задачею залишається програмна реалізація застосунку для автоматизації перевірки знань.

Б.2 Функціональне призначення

Система має реалізовувати наступні функціональні вимоги:

− реєстрація та автентифікація користувачів;

− завантаження та оновлення аватара користувача через налаштування профілю;

− створення та редагування тестів (вікторин);

− проходження тестів із відображенням результатів;

− пошук користувачів за іменами.

Б.3 Опис логічної структури

Для реалізації програмного забезпечення для автоматизації перевірки знань було обрано варіант реалізації у вигляді односторінкового додатку - SPA (Single Page Application).

Веб-сайт структуровано навколо п’яти ключових сторінкових компонентів:

− головна сторінка, що надає функціонал для пошуку користувачів;

− сторінка профілю користувача, де відображається список усіх пройдених тестів;

– сторінка керування налаштуваннями особистого профілю;

– сторінка для створення або редагування тестів (вікторин);

– сторінка безпосереднього проходження тесту.

Для забезпечення роботи додатку було розроблено API, розташоване на спеціально виділеному віддаленому сервері. Інтерфейс програмування застосунків (Application Programming Interface, API) являє собою набір методів, які розробник може використовувати для доступу до сховища даних системи.

Загальна схема функціонування програми представлена на рисунку Б.1.

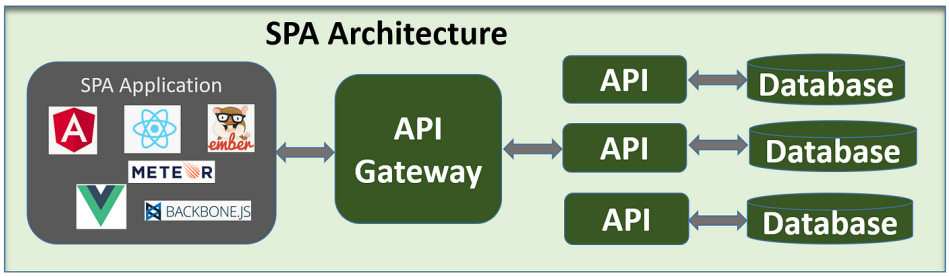

Рисунок Б.1 – Загальна структурна схема роботи програми

Під час розробки програмного забезпечення, призначеного для автоматизації тестування знань, було застосовано технологію REST (Representational State Transfer).

Взаємодія між клієнтами та серверами відбувається шляхом надсилання порцій даних, що називаються повідомленнями. Ті повідомлення, що ініціюються клієнтом, позначаються як "запити", тоді як ті, що надходять від сервера, іменуються "відповідями".

В застосунку передбачено чотири різновиди запитів:

– GET застосовується з метою отримання даних за певним запитом. Опрацювавши запит, GET повертає запитуваний ресурс, як правило, у форматі XML або JSON;

– POST використовується у ситуаціях, коли необхідно створити новий елемент;

– PATCH служить для часткової модифікації існуючого об'єкта;

– PUT призначений для повної заміни даних в уже наявному об'єкті;

– DELETE використовується для вилучення об'єкта, що існує.

Для здійснення комунікації із сервером було прийнято рішення використовувати бібліотеку Axios. Для керування навігацією в проєкті задіяна бібліотека «react-router». Схема маршрутизації в додатку для автоматизації перевірки знань зображено на рисунку Б.2.

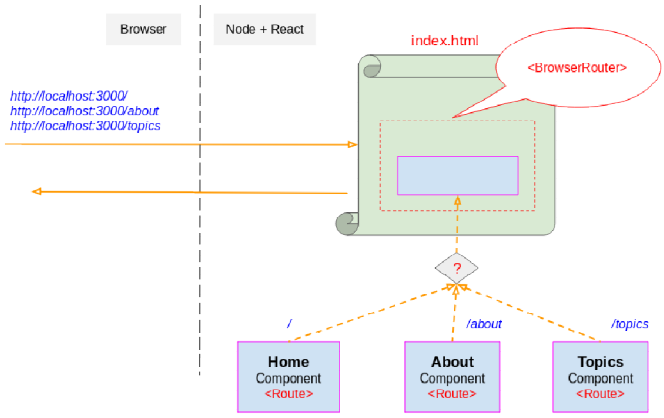

Рисунок Б.2 – Схема маршрутизації в додатку для автоматизації перевірки знань

Б.4 Використані технічні засоби

Для нормальної роботи веб-сайту та можливості його подальшого вдосконалення необхідно забезпечити мінімальні апаратні вимоги: персональний комп’ютер з процесором частотою не менше 2 ГГц, оперативною пам’яттю об’ємом щонайменше 4 ГБ та вільним дисковим простором не менше 5 ГБ, достатнім для встановлення браузера. Крім того, на комп’ютері має бути встановлено відповідне програмне забезпечення, зокрема операційна система та сучасний веб-браузер.

ДОДАТОК В

Текст програми

```
/*
============================================================
   QUIZ EDITOR FORM (React + React Hook Form)

============================================================
*/

     import React, { FC } from "react";
     import { useForm, FormProvider } from "react-hook-
form";
     import { yupResolver } from
"@hookform/resolvers/yup";
     import styled from "styled-components";
     import { useSelector } from "react-redux";

     import { validationSchema } from "./validation";
     import { IQuiz, IQuestion } from
"../../redux/interfaces";
     import { quizSelector, userSelector } from
"../../redux/selectors";
     import { AddQuestionItem, AddQuestionList,
PreviewImageUpload } from ".";

     /* Styled wrapper for layout */
     const Wrapper = styled.section`
       display: flex;
       flex-direction: column;
       align-items: center;
       margin-top: 16px;
     `;

     /* Submit button styles */
     const SubmitButton = styled.button`
       padding: 10px 18px;
       margin-bottom: 25px;
       background-color: #309d8f;
       border: none;
       border-radius: 14px;
       color: #fff;
       cursor: pointer;

       &:hover {
         background-color: #25796e;
       }
```

```
    `;

    /* Controls container */
    const ControlPanel = styled.div`
      display: flex;
      justify-content: center;
    `;

    /* Component props interface */
    interface QuizEditorProps {
      onSubmitQuiz: (quiz: IQuiz) => void;
      buttonLabel: string;
    }

    /* Form data structure */
    type QuizFormData = {
      title: string;
      questions: IQuestion[];
      image?: File;
    };

    /* Quiz editor component */
    const QuizEditor: FC<QuizEditorProps> = ({
onSubmitQuiz, buttonLabel }) => {
      const { currentQuiz } =
useSelector(quizSelector);
      const { currentUser } =
useSelector(userSelector);

      /* Initialize form with validation */
      const methods = useForm<QuizFormData>({
        resolver: yupResolver(validationSchema),
        defaultValues: currentQuiz || {
          title: "",
          questions: [],
          owner: currentUser.username,
        },
      });

      return (
        <FormProvider {...methods}>
          <form

onSubmit={methods.handleSubmit(onSubmitQuiz)}
            encType="multipart/form-data"
```

```
            autoComplete="off"
          >
            <Wrapper>
              {/* Quiz title and basic settings */}
              <AddQuestionItem type="quiz" />

              {/* Preview image upload */}
              <PreviewImageUpload />

              {/* Questions list */}
              <AddQuestionList />
            </Wrapper>

            <ControlPanel>
              <SubmitButton
type="submit">{buttonLabel}</SubmitButton>
            </ControlPanel>
          </form>
        </FormProvider>
      );
    };

    export default QuizEditor;

/*
============================================================
   REDUX API & ACTION CREATORS (Axios)

============================================================
*/
    import axios from "axios";
    import { ACTION_QUIZ, API_URI } from "../contants";
    import { IQuiz } from "../interfaces";
    import { logout, setUserProfile } from "./user";

    /* Axios instance configuration */
    const httpClient = axios.create({
      baseURL: `${API_URI}/api`,
    });

    /* Attach token to every request */
    httpClient.interceptors.request.use((config) => {
      const token = localStorage.getItem("token");
      if (token) {
        config.headers["x-access-token"] = token;
```

```
    }
    return config;
  });

  /* Quiz actions */
  export const loadQuizzes = (quizzes: IQuiz[]) => ({
    type: ACTION_QUIZ.SET_QUIZZES,
    payload: quizzes,
  });

  export const selectQuiz = (quiz: IQuiz) => ({
    type: ACTION_QUIZ.SET_CURRENT_QUIZ,
    payload: quiz,
  });

  export const beginQuiz = (quiz: IQuiz) => ({
    type: ACTION_QUIZ.START_QUIZ,
    payload: quiz,
  });

  export const markAnswered = (value: boolean) => ({
    type: ACTION_QUIZ.SET_IS_ANSWERED,
    payload: value,
  });

  export const saveAnswer = (answer: unknown) => ({
    type: ACTION_QUIZ.WRITE_ANSWER,
    payload: answer,
  });

  export const goToNextQuestion = () => ({
    type: ACTION_QUIZ.SET_NEXT_QUESTION,
  });

  export const resetQuiz = () => ({
    type: ACTION_QUIZ.CLEAR_QUIZ,
  });

  /* Async actions */
  export const requestAllQuizzes = () => async
(dispatch: any) => {
    try {
      const response = await
httpClient.get("/quizzes");
      dispatch(loadQuizzes(response.data.data));
```

```
      } catch (error) {
        dispatch(logout());
      }
    };

    export const requestUserQuizzes = (username:
string) => async (dispatch: any) => {
      try {
        const response = await
httpClient.get(`/quizzes/${username}`);
        dispatch(setUserProfile(response.data.user));
        dispatch(loadQuizzes(response.data.data));
      } catch (error) {
        dispatch(logout());
      }
    };

/*
============================================================
   USER REDUCER

============================================================
*/

    import { ACTION_USER } from "../contants";
    import { IUserState } from "../interfaces";
    import { AnyAction } from "../types";

    /* Default empty user object */
    const emptyUser = {
      username: "",
      avatar: "",
      email: "",
    };

    /* Initial state */
    const initialState: IUserState = {
      currentUser:
JSON.parse(localStorage.getItem("user") || "null") ||
emptyUser,
      answers: [],
      result: null,
      isAuthorized:
Boolean(localStorage.getItem("token")),
      profile: emptyUser,
```

```
};

/* User reducer */
const userReducer = (
  state = initialState,
  action: AnyAction
): IUserState => {
  switch (action.type) {
    case ACTION_USER.LOGGED_IN:
    case ACTION_USER.REGISTERED:
      return { ...state, isAuthorized: true };

    case ACTION_USER.LOGGED_OUT:
      return { ...state, isAuthorized: false };

    case ACTION_USER.SET_AVATAR:
      return {
        ...state,
        currentUser: { ...state.currentUser,
avatar: action.payload },
      };

    case ACTION_USER.SET_USER:
      return { ...state, currentUser:
action.payload };

    case ACTION_USER.SET_USER_PROFILE:
      return { ...state, profile: action.payload };

    case ACTION_USER.CLEAR_ANSWERS:
      return { ...initialState };

    case ACTION_USER.SET_RESULTS:
    case ACTION_USER.ADD_ANSWER:
      return { ...state, result: action.payload };

    default:
      return state;
  }
};

export default userReducer;
```

```
/*
============================================================
   USER SERVICE

============================================================
*/

     const bcrypt = require("bcryptjs");
     const UserModel = require("../models/user-model");
     const AuthService = require("./auth-service");
     const { HTTP_STATUS } = require("../constants");
     const { QueryError } =
require("../helpers/errorHandler");
     const { deleteFile } =
require("../helpers/fileSystem");

     /* Create and register new user */
     const registerUser = async ({ email, password,
username }) => {
       try {
         const hashedPassword = await
bcrypt.hash(password, 10);

         const user = new UserModel({
           email,
           username,
           password: hashedPassword,
         });
         user.token = await
AuthService.getSignedToken(user._id, username);
         await user.save();
         return user;
       } catch {
         throw new QueryError(HTTP_STATUS.BAD_REQUEST,
"User registration failed");
       }
     };

     /* Update user avatar */
     const changeUserAvatar = async (username,
avatarPath) => {
       const existingUser = await UserModel.findOne({
username });

       if (existingUser?.avatar) {
```

```
        deleteFile(existingUser.avatar);
      }
      await UserModel.updateOne({ username }, { avatar:
avatarPath });
    };

    /* Search users by username */
    const searchUsers = async (query) => {
      return UserModel.find({ username: new
RegExp(query, "i") });
    };

    /* Get public user data */
    const getPublicUserData = async (username) => {
      const user = await UserModel.findOne({ username
});
      if (!user) return null;

      return {
        username: user.username,
        avatar: user.avatar,
        email: user.email,
      };
    };

    module.exports = {
      registerUser,
      changeUserAvatar,
      searchUsers,
      getPublicUserData,
    };
```

ДОДАТОК Г

Слайди презентації

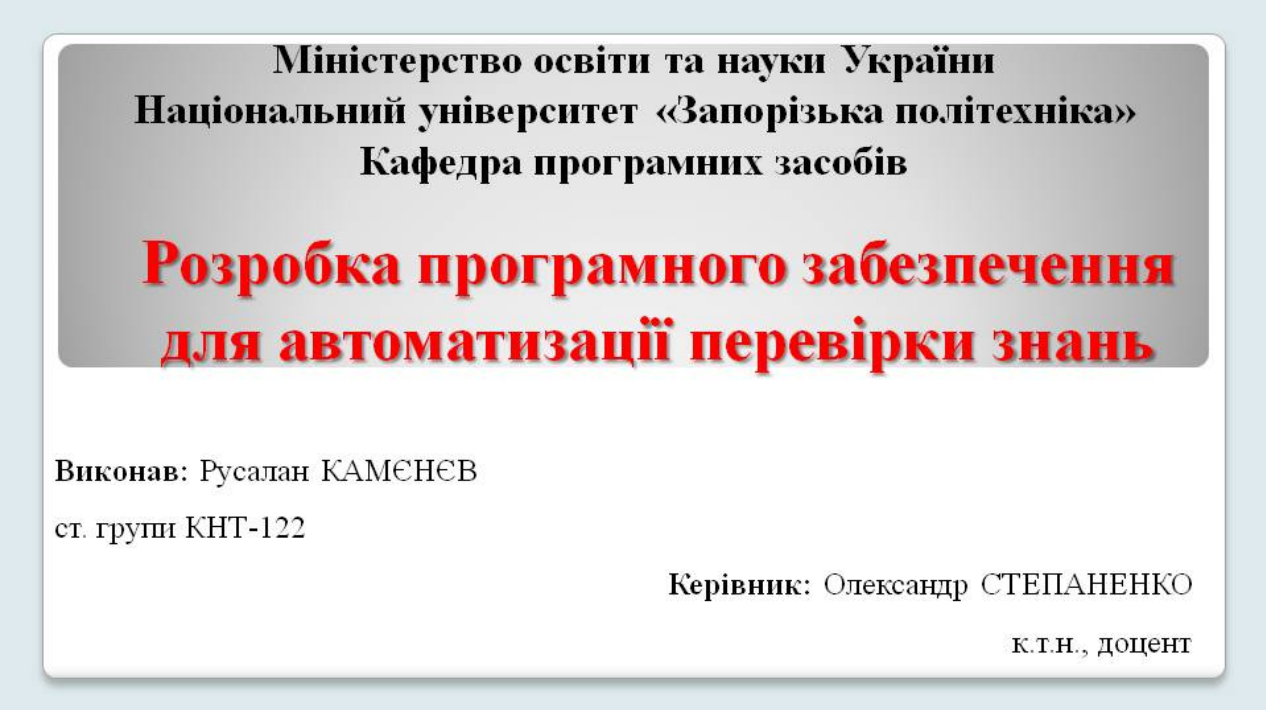

Рисунок Г.1 – Слайд 1

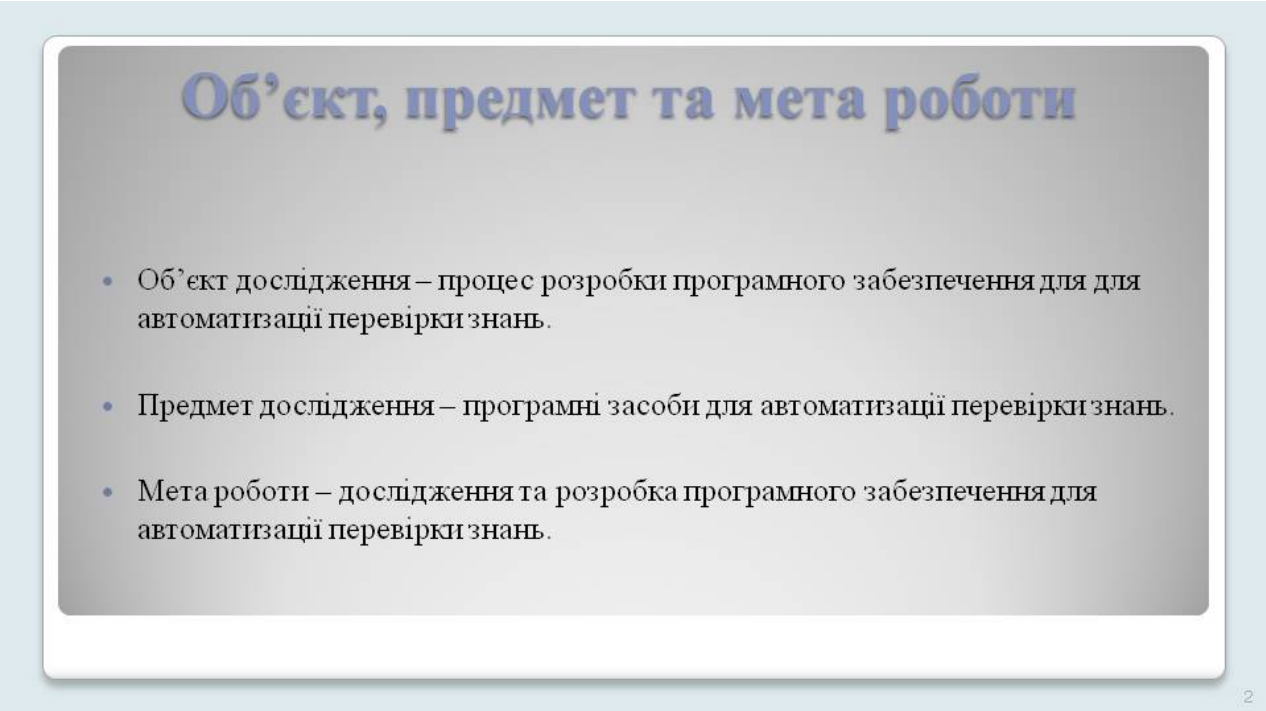

Рисунок Г.2 – Слайд 2

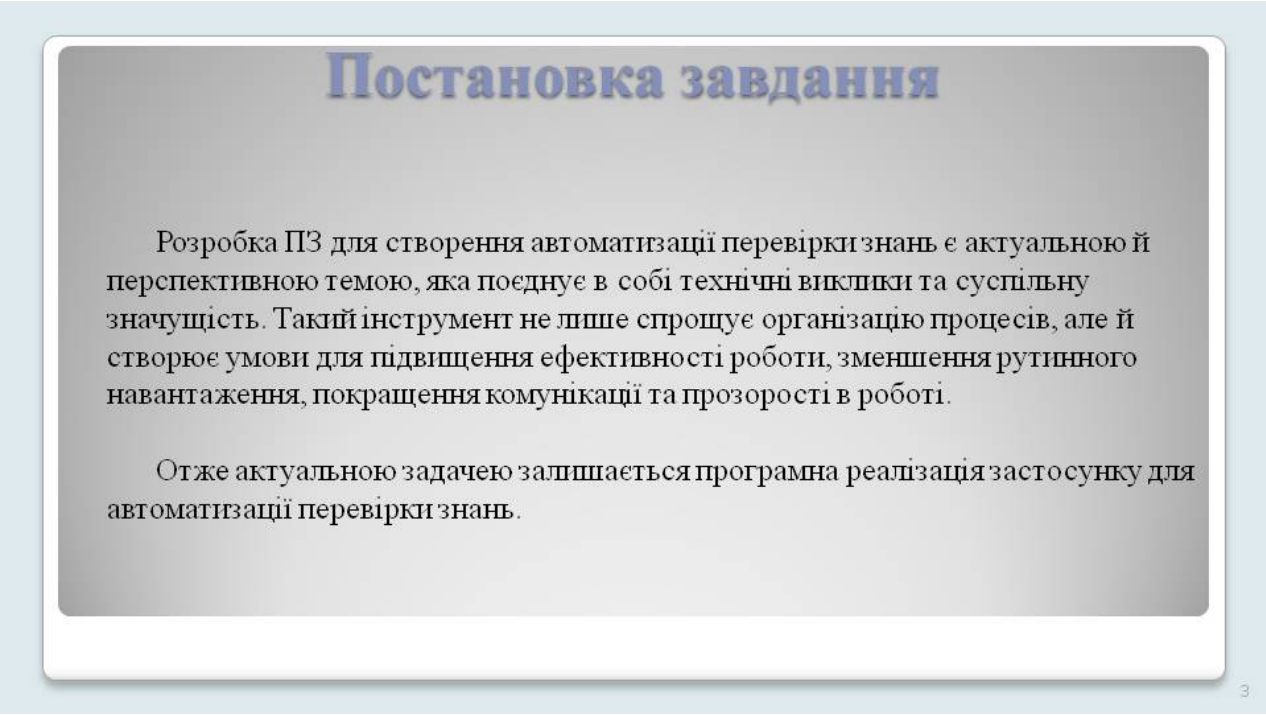

Рисунок Г.3 – Слайд 3

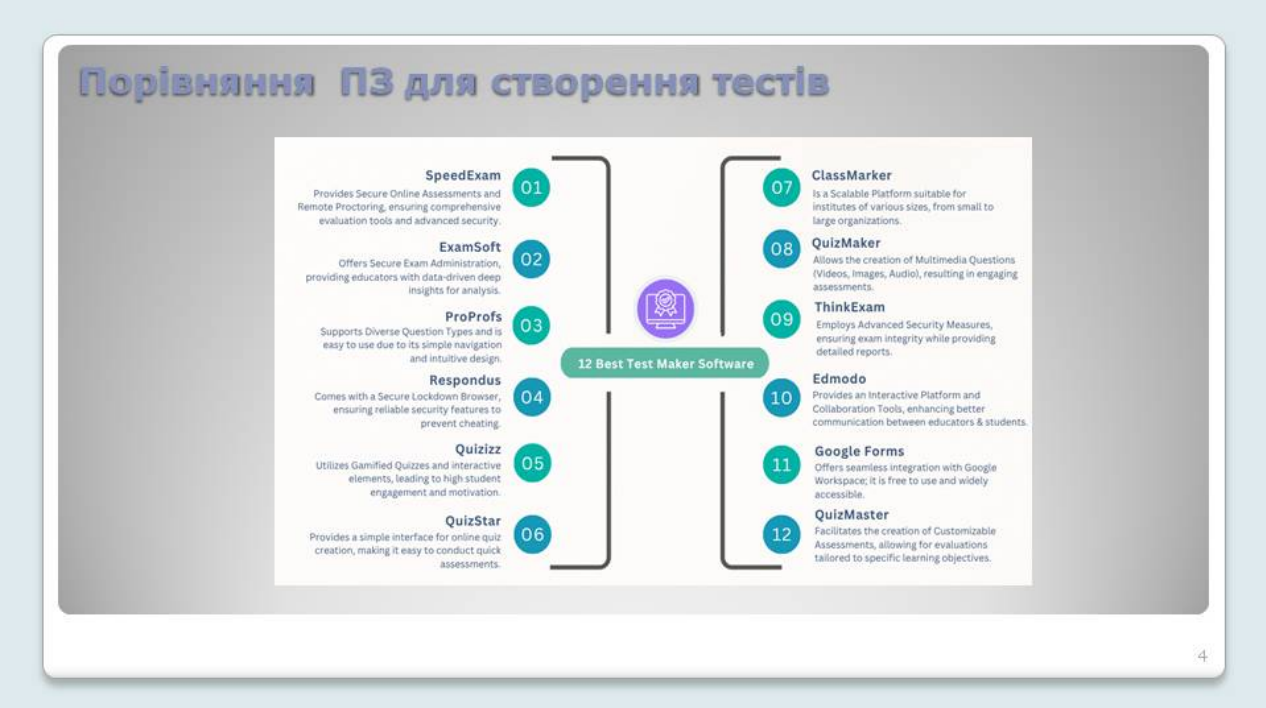

Рисунок Г.4 – Слайд 4

Рисунок Г.5 – Слайд 5

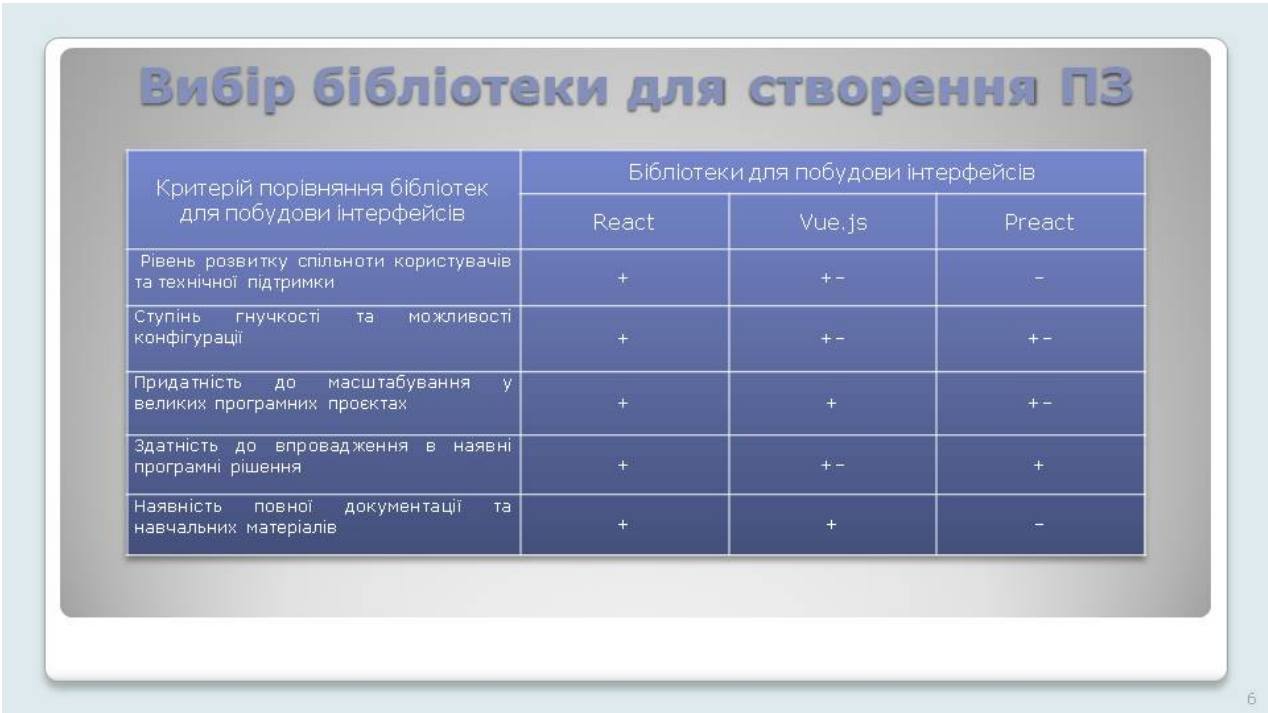

Рисунок Г.6 – Слайд 6

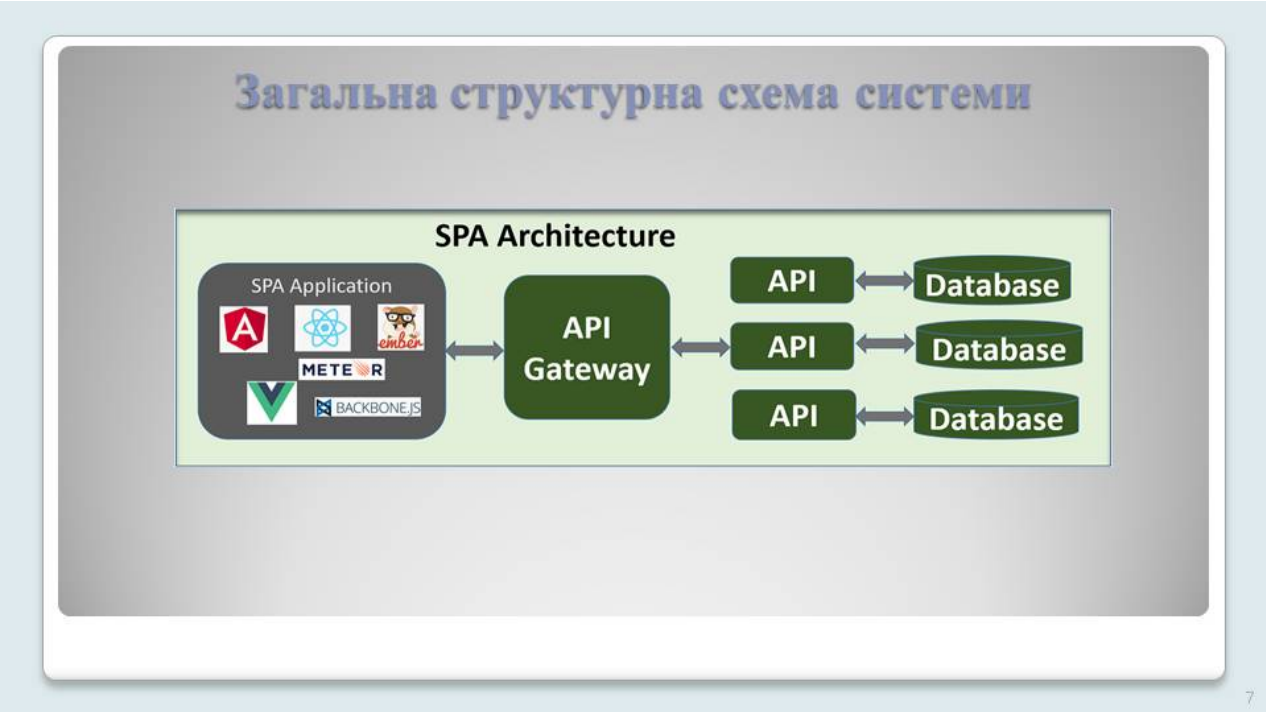

Рисунок Г.7 – Слайд 7

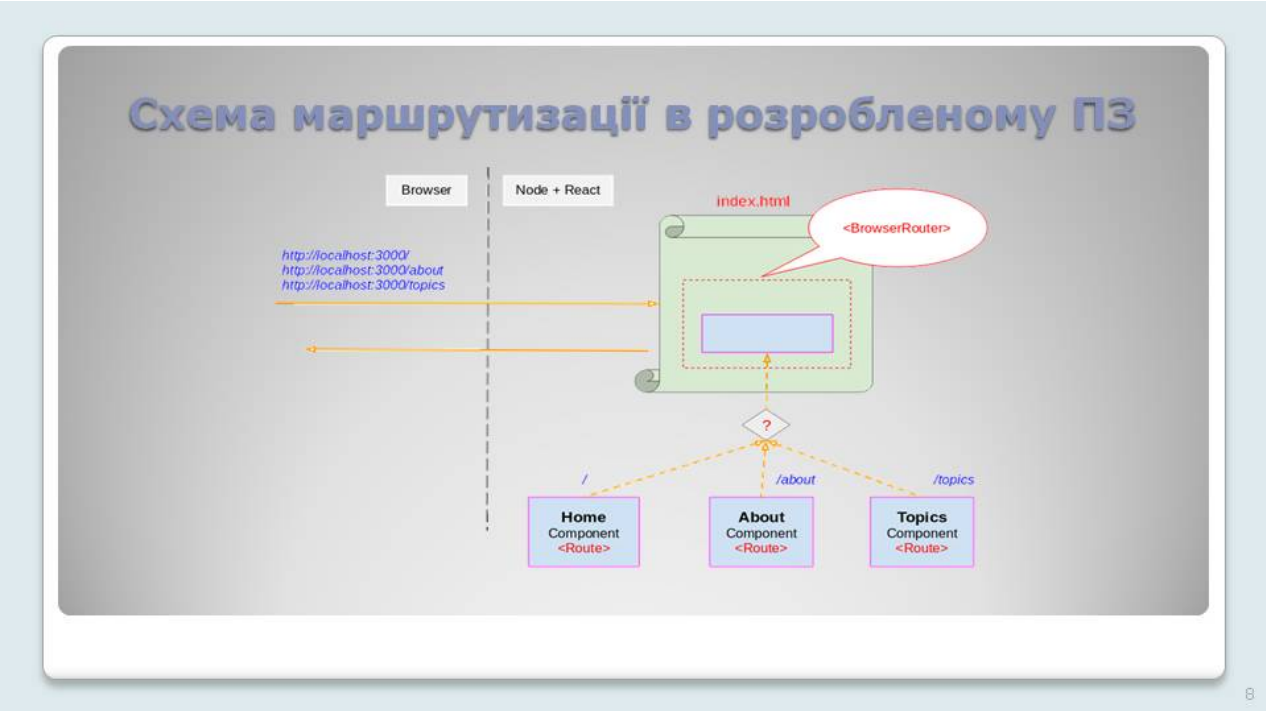

Рисунок Г.8 – Слайд 8

Рисунок Г.9 – Слайд 9

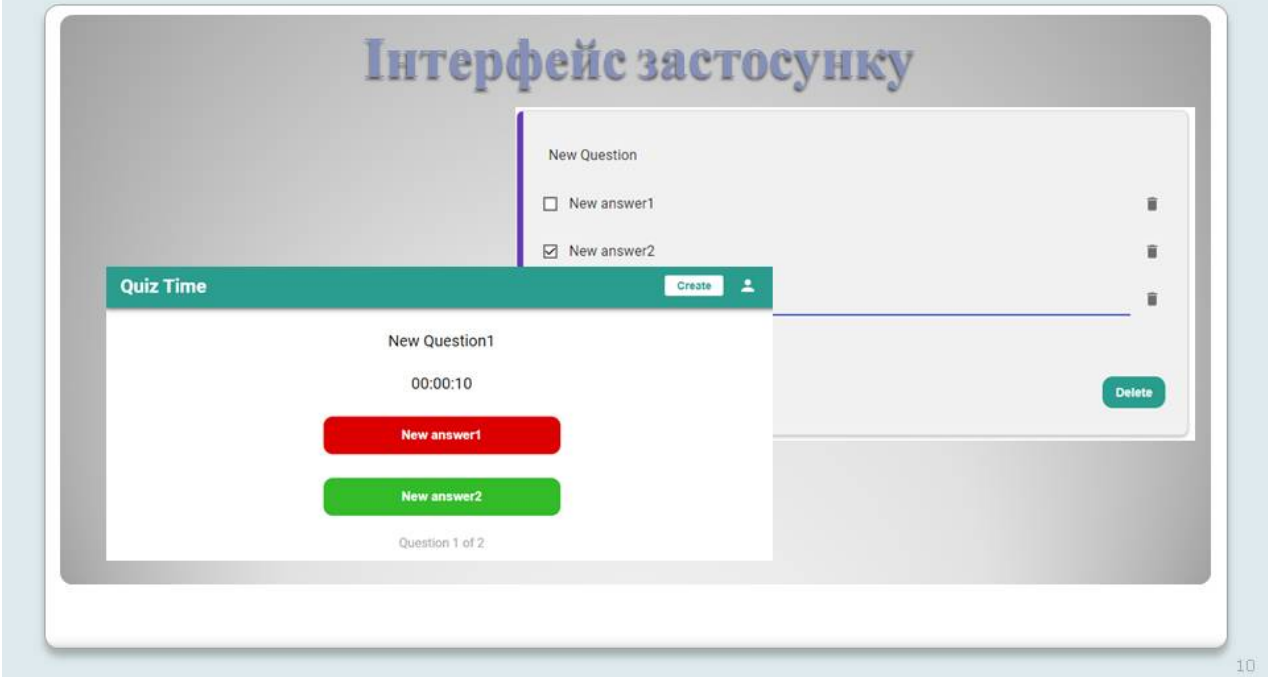

Рисунок Г.10 – Слайд 10

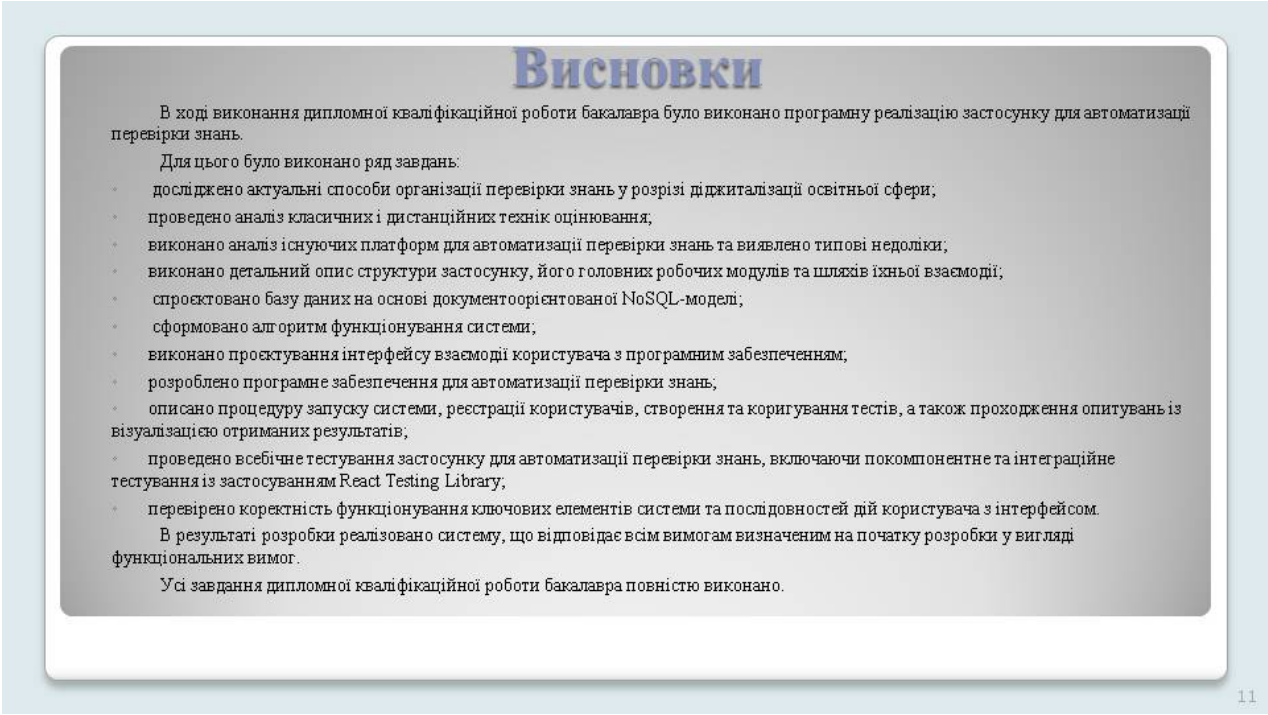

Рисунок Г.11 – Слайд 11