Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Конструктор вікторин
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-5
Антон Яблонський
(власне ім'я, прізвище)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)
Керівник ас. Дмитро Малєєв
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Яблонському Антону Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи Конструктор вікторин

затверджена наказом університету від 19 травня 20 25 р. № 378Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18 червня 20 25 р.

3. Вихідні дані до роботи офіційні документації до React.js, TypeScript, Firebase, OpenAI API, ShadCN UI, Vite, Auth0, Tailwind CSS, Redux Toolkit, Nest.js, Google Cloud, а також матеріали наукових статей із тематикою гейміфікованого навчання, тестування та інтеграції штучного інтелекту в освітні системи.

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної галузі та постановка задачі

2) Проектування системи

3) Програмна реалізація

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.05.2025	виконано
2	Визначення проблеми	20.05.2025	виконано
3	Аналіз існуючих аналогів	21.05.2025	виконано
4	Розробка компонентів системи	25.05.2025	виконано
5	Реалізація вебдодатку	01.06.2025	виконано
6	Оформлення пояснювальної записки	06.06.2025	виконано
7	Захист	18.06.2025	виконано

Дата видачі завдання 19 травня 20 25 р.

Здобувач _____
(підпис)

Керівник роботи _____ асистент Дмитро Малєєв
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 84 с., 20 рис., 1 дод., 21 джерело.

КВІЗ, ОСВІТНІЙ ВЕБДОДАТОК, ПЕРСОНАЛІЗАЦІЯ, ШТУЧНИЙ ІНТЕЛЕКТ, AI-GENERATED QUESTIONS, CLIENT-SERVER ARCHITECTURE, GPT-4, OPENAI API, QUIZ PLATFORM.

Об’єктом дослідження є сучасні вебдодатки, призначені для організації інтерактивного навчання у форматі вікторин.

Предметом дослідження виступають технології створення та реалізації системи для генерації, проходження й оцінювання тестових завдань з підтримкою штучного інтелекту.

Метою роботи є розробка вебдодатку для створення персоналізованих вікторин, що дозволяє користувачам формувати запитання вручну або автоматично за допомогою мовної моделі OpenAI.

У дослідженні використано методи аналізу літературних джерел, компаративного аналізу аналогів, проектування програмної архітектури, реалізації клієнт-серверної взаємодії та інтеграції API OpenAI.

У результаті реалізовано функціональний вебдодаток із підтримкою генерації запитань за темою, проходженням тестів і підрахунком балів, що забезпечує новий підхід до створення навчального контенту.

Розроблену систему рекомендовано для використання в освітніх середовищах та для самоосвіти.

ABSTRACT

Bachelor’s thesis contains: 84 pp., 20 fig., 1 ann., 21 references.

AI-GENERATED QUESTIONS, ARTIFICIAL INTELLIGENCE, CLIENT-SERVER ARCHITECTURE, EDUCATIONAL WEB APPLICATION, GPT-4, OPENAI API, PERSONALIZATION, QUIZ, QUIZ PLATFORM.

The object of the research is modern web applications designed for organizing interactive learning in the format of quizzes.

The subject of the research is the technologies for designing and implementing a system for generating, taking, and evaluating test tasks supported by artificial intelligence.

The aim of the work is to develop a web application for creating personalized quizzes, allowing users to formulate questions manually or automatically using the OpenAI language model.

The research employs methods such as literature analysis, comparative analysis of existing solutions, software architecture design, implementation of client-server interaction, and OpenAI API integration.

As a result, a functional web application was developed with support for topic-based question generation, test completion, and scoring, offering a novel approach to educational content creation.

The developed system is recommended for use in educational environments as well as for self-learning purposes.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AI – Artificial Intelligence – штучний інтелект;

API – Application Programming Interface – прикладний програмний інтерфейс;

CRM – Customer Relationship Management – управління взаємовідносинами з клієнтами;

HTTP – Hypertext Transfer Protocol – протокол передачі гіпертексту;

REST – Representational State Transfer – передача репрезентативного стану;

SDK – Software Development Kit – комплект засобів розробки;

SPA – Single Page Application – односторінковий застосунок;

SQL – Structured Query Language – мова структурованих запитів;

UI – User Interface – інтерфейс користувача;

UX – User Experience – користувацький досвід.

ВСТУП

У сучасних умовах стрімкого розвитку цифрових технологій усе більше уваги приділяється засобам інтерактивного навчання, зокрема системам тестування знань та вікторин. Ці інструменти активно впроваджуються в освітній, корпоративний і розважальний простір завдяки своїй універсальності, доступності й здатності залучати користувача до активної участі в процесі. Успіх таких рішень пояснюється тим, що вони поєднують елементи самоконтролю, мотивації та оцінювання у гнучкій, зрозумілій формі.

На тлі цих тенденцій зростає попит на рішення, які не лише забезпечують базовий функціонал створення вікторин, а й дозволяють автоматизувати створення контенту, проводити багатокористувацькі сесії, адмініструвати доступ і зберігати результати в структурованому вигляді. Особливої актуальності набувають технології штучного інтелекту, які відкривають нові можливості для генерації тестових завдань, адаптації рівня складності та персоналізації взаємодії з користувачем. Впровадження таких підходів дозволяє підвищити ефективність навчання, скоротити час на підготовку матеріалів та забезпечити індивідуальний підхід у процесі тестування.

На сьогодні відчувається потреба у розробці спеціалізованих вебдодатків, які могли б об’єднувати функціонал конструктора вікторин, можливість генерування контенту за допомогою штучного інтелекту, інструменти організації спільного проходження тестів, а також системи підрахунку балів і адміністрування сесій. Особливо важливою є реалізація простого та інтуїтивного інтерфейсу, який дозволить користувачеві без спеціальних знань швидко створювати вікторини, змінювати їхню структуру, запускати сесії та отримувати зведену аналітику. Таке рішення може бути корисним як для освітніх установ – шкіл, вишів, курсів, так і для

внутрішнього навчання у компаніях або створення інтерактивного контенту для широкої аудиторії.

Актуальність цієї роботи зумовлена необхідністю створення відкритого, адаптивного та функціонального рішення, яке дозволяло б не лише формувати вікторини, а й реалізовувати повноцінну систему керування контентом і процесом взаємодії з користувачами. Застосування штучного інтелекту у цьому контексті дозволяє вивести традиційні системи тестування на новий рівень, спрощуючи підготовку матеріалів та забезпечуючи їх динамічну адаптацію до потреб користувача. В умовах зростаючої кількості онлайн-курсів, самостійного навчання та корпоративних тренінгів універсальні системи створення вікторин з автоматичною генерацією контенту набувають особливої значущості.

Метою цієї кваліфікаційної роботи є розробка вебдодатку типу конструктора вікторин, який дозволяє створювати різні типи тестових завдань, запускати їх для одночасного проходження у вигляді сесій, автоматично підраховувати бали, формувати зведення результатів, а також інтегрувати штучний інтелект для генерації запитань. Розроблене рішення повинно відповідати сучасним вимогам до зручності, адаптивності та функціональної повноти. Основними сферами застосування є освіта, онлайн-курси, внутрішнє корпоративне навчання, маркетингові кампанії, опитування та розважальні заходи, де потрібне швидке та якісне тестування з можливістю аналітики.

Таким чином, запропонована система має на меті заповнити прогалину між простими формами онлайн-опитування та складними корпоративними платформами, об’єднуючи в собі інтуїтивний інтерфейс, гнучкий конструктор, функції багатокористувацької взаємодії та генерацію навчального контенту на основі штучного інтелекту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Вікторини як сучасний інструмент навчання та розваги

Вікторини давно зарекомендували себе як ефективний інструмент перевірки знань, самонавчання та взаємодії з аудиторією. Їх застосовують у багатьох сферах – від формальної освіти до маркетингових кампаній. Завдяки своїй простоті, динамічності та гнучкості, вікторини залишаються актуальними як у цифровому, так і в традиційному середовищі.

З появою вебтехнологій та мобільних платформ створення і проходження вікторин стало доступним будь-де і будь-коли. Сучасні користувачі очікують зручного інтерфейсу, швидкого доступу до контенту та можливості гнучко налаштовувати формат питань. Це призвело до появи нових сервісів, які пропонують розширений функціонал для створення, поширення та аналізу вікторин.

Особливої популярності набули інтерактивні вікторини, які не лише перевіряють знання, а й активно залучають учасників до процесу. Такі вікторини поєднують навчальні та розважальні елементи, використовуючи сучасні ІТ-рішення та механізми штучного інтелекту. Завдяки цьому цифрові вікторини стають універсальним інструментом для реалізації інформативного, адаптивного та цікавого контенту.

1.1.1 Інтерактивні системи тестування знань

Інтерактивні системи тестування знань стали невід’ємною складовою сучасного навчального процесу. Вони забезпечують швидкий зворотний зв’язок, автоматичне оцінювання результатів і гнучкість у налаштуванні формату завдань. Завдяки цьому такі системи активно використовуються в школах, університетах, корпоративному навчанні та під час сертифікації.

Різноманітність типів запитань – від одного вибору до відкритих відповідей – дозволяє адаптувати тести до цілей викладача або організатора. Інтерактивність полягає не лише у формі подачі матеріалу, а й у можливості миттєвої перевірки, повторного проходження або отримання підказок. Це сприяє кращому засвоєнню інформації та підвищує мотивацію користувачів.

Завдяки розвитку вебтехнологій тестування вийшло за межі класичних комп'ютерних програм і стало частиною вебсередовища. Системи стали доступними з будь-якого пристрою з інтернет-з'єднанням, що значно розширило коло користувачів і спростило інтеграцію з іншими цифровими освітніми інструментами.

Окрему роль відіграють механізми аналітики, які надають викладачам або адміністраторам змогу аналізувати результати в реальному часі. Це дозволяє не лише перевіряти рівень знань, а й виявляти слабкі місця у програмі навчання, адаптуючи її до потреб аудиторії. Таким чином, інтерактивне тестування стає потужним інструментом контролю та вдосконалення освітнього процесу.

1.1.2 Еволюція цифрових вікторин в освітніх і розважальних середовищах

Цифрові вікторини пройшли довгий шлях розвитку – від простих текстових тестів до складних інтерактивних платформ з мультимедійним супроводом. Спершу такі системи були локальними додатками, обмеженими за функціональністю і доступними лише на окремих пристроях. З розвитком інтернету і технологій вебпрограмування вікторини стали частиною хмарних сервісів, що зробило їх доступними для масового використання.

Однією з ключових тенденцій стала інтеграція вікторин у навчальні платформи, системи дистанційного навчання та корпоративні системи

управління знаннями. Це дозволило використовувати їх не лише як самостійний інструмент, а й як допоміжний засіб у комплексних освітніх рішеннях. Водночас вікторини почали виконувати не лише навчальну, а й мотиваційну функцію, зокрема у форматі гейміфікованих завдань.

Значну роль у популяризації вікторин відіграли мобільні додатки та адаптація платформ до смартфонів і планшетів. Це дозволило проводити тестування у зручному для користувача форматі – у транспорті, вдома або навіть під час живих заходів. Зросла і швидкість обробки результатів, що підвищило ефективність застосування вікторин у режимі реального часу.

Розвиток цифрових вікторин також супроводжувався змінами у дизайні користувацького інтерфейсу. Сучасні системи приділяють велику увагу зручності, візуальній привабливості та інтуїтивній навігації. Це сприяє зниженню порогу входу для нових користувачів і підвищенню рівня залучення до процесу тестування.

1.1.3 Використання вікторин як інструменту залучення

Вікторини активно використовуються як засіб залучення аудиторії в освітніх, комерційних і розважальних середовищах. Завдяки своїй ігровій природі вони здатні утримувати увагу користувача довше, ніж традиційні форми взаємодії. Це робить їх ефективним інструментом не лише для перевірки знань, а й для підтримки інтересу та мотивації.

У навчальному процесі вікторини сприяють активному засвоєнню матеріалу, оскільки залучають студента до безпосередньої участі. Миттєвий зворотний зв’язок, система балів або рейтингів, можливість проходити тести повторно – усе це стимулює учасників до вдосконалення своїх результатів. Такий формат сприяє не лише кращому запам’ятовуванню інформації, а й формуванню позитивного ставлення до навчання.

У корпоративному та маркетинговому середовищі вікторини часто застосовуються для взаємодії з клієнтами або працівниками. Вони можуть

бути частиною рекламних кампаній, інструментом оцінювання персоналу або засобом внутрішнього навчання. У всіх цих випадках вікторини допомагають створити інтерактивний досвід, що підвищує лояльність та інтерес до платформи або бренду.

1.1.4 Інформаційні технології для створення сучасних інтерактивних опитувань

Інформаційні технології відіграють ключову роль у створенні сучасних інтерактивних вікторин. Завдяки розвитку вебфреймворків, хмарних сервісів та мобільних платформ стало можливим реалізовувати функціонально насичені, масштабовані та зручні для користувача системи. Використання баз даних, REST API та клієнтських бібліотек дозволяє будувати динамічні інтерфейси та забезпечувати обробку великої кількості відповідей у режимі реального часу.

Системи вікторин дедалі частіше інтегруються з іншими платформами – системами управління навчанням, CRM, аналітичними модулями. Це дозволяє використовувати вікторини не як ізольований інструмент, а як частину загального цифрового середовища. Такі технологічні можливості сприяють персоналізації досвіду користувача, автоматизації звітності та підвищенню ефективності навчальних і бізнес-процесів.

1.1.5 Роль штучного інтелекту у генерації контенту

Штучний інтелект відкриває нові можливості у створенні контенту для вікторин, автоматизуючи процес формування запитань і відповідей. Використання моделей обробки природної мови дає змогу генерувати питання на основі вхідної тематики, враховуючи складність, контекст та

бажану форму. Це значно скорочує час підготовки матеріалів і дозволяє створювати унікальні варіанти тестів для кожного користувача.

Крім генерації питань, штучний інтелект може бути використаний для адаптації рівня складності вікторини в залежності від попередніх результатів учасника. Це сприяє індивідуалізації навчального процесу, підвищенню зацікавленості та мотивації. Такі інтелектуальні можливості поступово стають стандартом для сучасних цифрових освітніх платформ.

1.2 Стан ринку інструментів для створення вікторин

Ринок інструментів для створення вікторин активно розвивається і охоплює широкий спектр рішень – від простих форм з обмеженим функціоналом до комплексних інтерактивних платформ з елементами гейміфікації, аналітики та штучного інтелекту. Конкуренція на цьому ринку стимулює впровадження нових технологій і постійне вдосконалення функцій.

Сучасні платформи для створення вікторин орієнтовані на різні цільові аудиторії: освітні заклади, бізнес, маркетинг, а також індивідуальних користувачів. Кожен із сегментів має свої специфічні потреби, що впливає на підхід до реалізації функціоналу, інтерфейсу та моделі монетизації. Це зумовлює існування різноманітних рішень, які відрізняються за складністю, вартістю та можливостями налаштування.

Загальні тенденції на ринку включають впровадження адаптивного навчання, використання мобільних технологій, інтеграцію з іншими цифровими системами, а також застосування AI для генерації контенту. Однак поряд з прогресом залишаються і певні виклики, пов’язані з безпекою, конфіденційністю даних та сумісністю з існуючими інфраструктурами.

1.2.1 Основні гравці ринку онлайн-вікторин

Серед основних гравців ринку онлайн-вікторин можна виділити кілька платформ, які здобули значну популярність завдяки поєднанню доступності, функціональності та зручного інтерфейсу. Найбільш відомими є такі сервіси як Kahoot!, Quizizz, Google Forms, Typeform, Mentimeter та інші. Кожен з них має власну специфіку та орієнтований на певний сегмент користувачів.

Kahoot! (рисунок 1.1) спеціалізується на гейміфікованих вікторинах, що активно використовуються у навчальному процесі на різних рівнях – від шкільної освіти до корпоративного навчання. Його основною перевагою є можливість створювати інтерактивні тести з моментальним зворотним зв’язком, що підвищує залучення користувачів та сприяє кращому запам’ятовуванню інформації. Завдяки вбудованій системі балів та лідерборду, платформа стимулює елемент змагання, що позитивно впливає на мотивацію учасників. Платформа дозволяє проводити вікторини в реальному часі, що робить її ідеальною для живих занять, тренінгів, інтерактивних лекцій та онлайн-конференцій.

Рисунок 1.1 – Лого Kahoot!

Quizizz (рисунок 1.2) є схожим за функціональністю, але має акцент на асинхронне проходження тестів. Учасники можуть самостійно виконувати завдання у зручний час, а організатор отримує повну аналітику результатів. Платформа підтримує інтеграцію з Google Classroom та іншими освітніми сервісами, що підвищує її популярність серед викладачів.

Рисунок 1.2 – Лого Quizizz

Google Forms (рисунок 1.3) виступає як універсальний інструмент для створення форм і опитувань, включаючи тести з автоматичним оцінюванням. Його основна перевага – простота у використанні, безкоштовність та тісна інтеграція з іншими сервісами Google. Однак за функціональністю ця платформа поступається спеціалізованим рішенням.

Рисунок 1.3 – Лого Google Forms

Typeform (рисунок 1.4) орієнтований переважно на створення опитувань з високим рівнем дизайну та зручності. Його фокус – на UX, тому він часто використовується у маркетингових і дослідницьких проектах. Mentimeter, у свою чергу, пропонує можливості для інтерактивних презентацій, які включають питання, голосування та вікторини, що робить його корисним для виступів і тренінгів.

Рисунок 1.4 – Лого Typeform

Крім цих міжнародних рішень, на ринку існують і локальні платформи, які орієнтовані на національні освітні системи або внутрішні корпоративні потреби. Їх перевага полягає у кращій відповідності локальним вимогам, мові інтерфейсу та особливостям нормативної бази. Таким чином, ринок онлайн-вікторин є досить різноманітним і динамічним, з великою кількістю конкурентів, кожен з яких займає власну нішу.

1.2.2 Сегментація ринку за типами платформ

Ринок інструментів для створення вікторин можна умовно поділити на кілька основних сегментів залежно від типу платформи. Найбільш поширеними є хмарні вебсервіси, мобільні додатки, локальні десктопні програми та інтегровані модулі в межах освітніх систем. Кожен із цих форматів має свої особливості, переваги та недоліки.

Хмарні платформи є найпопулярнішими завдяки доступності з будь-якого пристрою, автоматичному збереженню даних і можливості колективної роботи. Користувачі можуть створювати, запускати та аналізувати вікторини без потреби в локальній інсталяції програмного забезпечення. До таких рішень належать Google Forms, Kahoot!, Quizizz, Typeform та інші. Вони ідеально підходять для дистанційного навчання, онлайн-подій та масштабних опитувань.

Мобільні додатки, як правило, орієнтовані на кінцевого користувача, який проходить вікторини у зручний для себе час. Додатки можуть працювати як самостійно, так і в парі з вебверсіями платформ. Їх перевагою є зручність доступу, сповіщення про нові тести та можливість інтеграції з іншими мобільними сервісами. Такі додатки особливо популярні серед учнів, які користуються смартфонами для навчання.

Локальні десктопні рішення мають обмежене поширення, але все ще використовуються в організаціях, де доступ до інтернету є нестабільним або обмеженим. Вони часто пропонують розширені функції без потреби у вебз'єднанні, що може бути критичним у певних галузях, зокрема в навчальних закладах з обмеженою інфраструктурою.

Окрему нішу займають модулі вікторин, які інтегруються у великі освітні платформи, такі як Moodle, Blackboard або Microsoft Teams. Вони зазвичай призначені для формального навчання, підтримують складні структури курсів, календарі подій, оцінювання та інші можливості для викладачів. Їх використання передбачає глибшу інтеграцію у навчальний процес.

Таким чином, сегментація ринку за типами платформ дозволяє користувачам обрати найбільш зручний і функціональний варіант відповідно до конкретних завдань. Вибір залежить від рівня технічної підготовки користувача, наявності інтернету, цілей тестування та особливостей аудиторії.

1.2.3 Порівняльний аналіз функціональності

Функціональні можливості платформ для створення вікторин можуть суттєво відрізнятися залежно від їхньої цільової аудиторії та концепції. Одні сервіси орієнтовані на швидке створення базових форм з обмеженим налаштуванням, тоді як інші пропонують складні конструктори з підтримкою мультимедійних елементів, гейміфікації, аналітики та адаптивного навчання. Така різноманітність забезпечує користувачам широкий вибір відповідно до їхніх потреб і рівня підготовки.

Ключовими параметрами для порівняння є типи запитань, можливість автоматичного оцінювання, наявність шаблонів, інтеграція з іншими сервісами, підтримка командної роботи та аналітичні інструменти. Наприклад, Google Forms пропонує базові функції зручного створення тестів, але обмежує візуальні та інтерактивні елементи. У той час як Kahoot! та Quizizz дозволяють створювати яскраві, гейміфіковані вікторини з рейтингами, таймерами та візуальними ефектами.

Також важливо враховувати, наскільки платформа підтримує кастомізацію та адаптацію контенту під різні сценарії використання. Деякі сервіси дозволяють створювати складні гілки логіки, налаштовувати доступи та працювати з великими обсягами учасників. Порівняльний аналіз таких функцій дозволяє краще зрозуміти, які інструменти підходять для конкретного освітнього, корпоративного чи дослідницького завдання.

1.2.4 Виявлені переваги та недоліки існуючих платформ

Кожна з платформ для створення вікторин має свої сильні сторони, які формують її конкурентну перевагу на ринку. До основних переваг популярних рішень можна віднести простоту у використанні, доступність з різних пристроїв, широкий вибір типів запитань та інтеграцію з іншими

сервісами. Крім того, деякі платформи забезпечують візуально привабливий інтерфейс, що підвищує інтерес до проходження вікторин.

Важливою перевагою є також наявність аналітичних інструментів. Платформи, які дозволяють відстежувати прогрес користувачів, генерувати звіти та оцінювати ефективність тестування, мають вищу цінність для освітніх установ та бізнесу. Такі функції дозволяють оптимізувати навчальні програми або внутрішнє корпоративне навчання на основі реальних даних.

Серед недоліків часто згадується обмежена кастомізація, коли користувач не може змінити логіку проходження або оформлення тесту. Також проблемою може бути недостатній рівень локалізації – відсутність підтримки деяких мов або особливостей локального ринку. Деякі сервіси накладають обмеження у безкоштовних версіях, що знижує їхню привабливість для індивідуальних користувачів або невеликих організацій.

Ще одним обмеженням є відсутність адаптивного навчання у більшості платформ. Не всі сервіси підтримують динамічну зміну рівня складності або персоналізовані маршрути проходження. Це знижує ефективність тестування у випадках, коли важлива індивідуальна траєкторія користувача. Незважаючи на ці обмеження, більшість платформ продовжують активно розвиватися, поступово розширюючи функціонал і долаючи виявлені недоліки.

1.2.5 Тенденції та інновації в галузі

Сфера створення онлайн-вікторин постійно оновлюється під впливом технологічного прогресу та змін у поведінці користувачів. Серед ключових тенденцій останніх років можна відзначити гейміфікацію, адаптивне навчання, інтеграцію з соціальними мережами та мобільну оптимізацію. Вікторини дедалі частіше стають частиною комплексних цифрових рішень, поєднуючи навчальні, мотиваційні та аналітичні компоненти.

Інноваційним напрямом є впровадження штучного інтелекту для автоматичної генерації питань, персоналізації тестів та аналізу поведінки користувачів. Також зростає інтерес до використання голосових та візуальних інтерфейсів, інтеграції з AR і VR технологіями для створення занурювального досвіду. Такі підходи дозволяють зробити процес тестування більш гнучким, інтерактивним і ефективним у різних контекстах.

1.2.6 Моделі монетизації та ціноутворення

Моделі монетизації платформ для створення вікторин залежать від їхньої цільової аудиторії, функціональності та ринкової стратегії. Найпоширенішою є freemium-модель, коли базові функції доступні безкоштовно, а розширені можливості надаються за плату. Це дозволяє залучити широку аудиторію, водночас стимулюючи користувачів перейти на преміум-версію заради додаткових функцій.

Іншою популярною моделлю є підписка, яка передбачає щомісячну або щорічну оплату за доступ до платформи. Такий підхід часто використовують у корпоративному секторі або в освітніх установах, де важлива стабільність, підтримка та регулярні оновлення. У деяких випадках також пропонуються разові платежі за певний функціонал або кількість користувачів.

Додатковим джерелом доходу для платформ можуть бути партнерські програми, реклама або інтеграція з іншими платними сервісами. Наприклад, платформи можуть пропонувати брендування інтерфейсу, розміщення логотипів замовника чи доступ до спеціальних аналітичних модулів. Такий гнучкий підхід до монетизації дозволяє адаптуватися до потреб різних сегментів ринку.

1.2.7 Проблеми та виклики ринку

Попри активний розвиток ринку, платформи для створення вікторин стикаються з низкою викликів. Однією з головних проблем є забезпечення конфіденційності даних користувачів, особливо у випадках, коли платформи обробляють особисту інформацію учасників. Це вимагає відповідності сучасним стандартам безпеки, зокрема міжнародним нормам захисту персональних даних.

Ще одним викликом є сумісність з іншими цифровими системами та адаптація до швидко змінюваних технологічних умов. Не всі платформи підтримують інтеграцію з навчальними середовищами, CRM чи хмарними сервісами, що обмежує їхнє використання в організаціях. Також актуальною залишається проблема мовної підтримки, доступності для людей з інвалідністю та забезпечення стабільної роботи при високому навантаженні.

1.3 Аналіз цільової аудиторії та її потреб

Ефективне проектування системи вікторин неможливе без урахування особливостей цільової аудиторії. Користувачі мають різний досвід, мотивацію та технічні навички, тому система повинна бути зручною, інтуїтивною та функціонально гнучкою. Розуміння потреб користувачів дозволяє створити рішення, яке відповідатиме реальним сценаріям використання.

До ключових груп користувачів зазвичай належать організатори вікторин і безпосередні учасники. Організатори очікують широких можливостей для налаштування контенту, керування доступом та аналізу результатів. Водночас учасники цінують простоту інтерфейсу, швидкий доступ до завдань і зрозумілий зворотний зв’язок після проходження тесту.

Крім функціональних вимог, важливу роль відіграє загальний користувацький досвід. До нього належить візуальна привабливість, логіка

навігації, стабільність роботи системи та підтримка з боку розробників. Забезпечення балансу між технічною складністю та простотою використання є визначальним фактором у створенні якісного продукту.

1.3.1 Класифікація користувачів онлайн-вікторин

У системах онлайн-вікторин користувачі можуть виконувати різні ролі, кожна з яких має свої цілі, завдання та очікування. Умовно їх можна поділити на дві основні групи: організатори та учасники. Такий поділ дозволяє точніше формулювати вимоги до функціональності, інтерфейсу та логіки взаємодії з системою.

Організатори вікторин – це викладачі, тренери, адміністратори курсів або менеджери, які створюють зміст, керують процесом тестування та аналізують результати. Вони зацікавлені у наявності гнучкого конструктора питань, можливості налаштування доступу, формування звітів та моніторингу активності користувачів. Для цієї категорії ключовими є зручність керування контентом і прозорість статистики.

Учасники вікторин – це студенти, працівники, клієнти або просто користувачі, які проходять тести для навчання, самоперевірки або розваги. Вони очікують зрозумілий інтерфейс, швидкий запуск завдань, зручне введення відповідей і миттєвий результат. Важливо також, щоб система була адаптована до різних пристроїв, включно зі смартфонами і планшетами.

Крім основних груп, у деяких випадках можуть бути задіяні й додаткові ролі, наприклад, адміністратори системи або технічна підтримка. Їхнє завдання – забезпечення працездатності платформи, контроль за дотриманням політик безпеки та конфіденційності, інтеграція з іншими сервісами. Таким чином, класифікація користувачів є необхідним етапом у проектуванні середовища для створення і проходження вікторин.

1.3.2 Потреби організаторів вікторин

Організатори вікторин очікують від системи зручного та швидкого процесу створення запитань. Для них важливо мати інтерфейс, який дозволяє додавати різні типи завдань, комбінувати варіанти відповідей, вставляти зображення або медіафайли, а також зберігати шаблони для повторного використання. Гнучкий редактор запитань з можливістю налаштування логіки проходження тесту є однією з основних потреб цієї групи користувачів.

Ще одним важливим аспектом є контроль над доступом до вікторин. Організатори мають потребу в інструментах, які дозволяють обмежити час виконання, встановлювати дедлайни, задавати кількість спроб або створювати захищені кімнати для проходження тестування. Також необхідно мати функції для призначення вікторин конкретним групам користувачів або індивідуальним учасникам.

Не менш значущою є можливість аналітики та оцінювання результатів. Організатори прагнуть бачити не лише загальні бали, а й детальну інформацію про відповіді, середній час виконання, рівень складності окремих завдань. Ці дані дають змогу коригувати навчальні матеріали, адаптувати складність і формувати об'єктивну картину успішності учасників.

1.3.3 Потреби учасників вікторин

Учасники вікторин зацікавлені у простому, інтуїтивно зрозумілому інтерфейсі, який дозволяє швидко почати виконання завдань без попередньої підготовки. Вони очікують, що проходження вікторини буде зручним як з комп’ютера, так і з мобільного пристрою. Важливими є чітка навігація, зрозуміла структура запитань і зручність введення відповідей.

Швидкий зворотний зв’язок також є важливою потребою для учасника. Після завершення тесту користувач хоче миттєво дізнатися результати, побачити правильні відповіді, отримати пояснення або підказки. Така інформація дозволяє оцінити свої знання, знайти помилки та, за потреби, повторити проходження для кращого результату.

Ще однією актуальною вимогою є гнучкість у доступі до вікторин. Учасники цінують можливість проходити завдання у зручний для них час, мати кілька спроб або перерву між питаннями. Для деяких користувачів важливою є підтримка різних мов, доступність для людей з порушеннями зору або слуху, а також стабільна робота платформи незалежно від технічних характеристик пристрою.

1.3.4 Вимоги до функціональності та інтерфейсу

Функціональність і зручність інтерфейсу є критичними факторами для обох основних категорій користувачів – як організаторів, так і учасників вікторин. Інтерфейс повинен бути логічно структурованим, мінімалістичним та зрозумілим без потреби в додаткових інструкціях. Особливо важливо забезпечити швидкий доступ до основних функцій та уникати перевантаження екрана зайвими елементами.

Серед функціональних вимог користувачі виділяють підтримку різних типів запитань, включаючи запитання з одним або кількома варіантами відповіді, відкриті запитання, а також запитання з використанням штучного інтелекту. Важливою є також можливість додавання медіаконтенту – зображень, аудіо чи відео, – що дозволяє урізноманітнити формат тестів і зробити їх більш наочними. Більшість користувачів очікують наявність таймерів, обмеження кількості спроб, збереження прогресу та автоматичного оцінювання результатів, що значно підвищує зручність і об’єктивність у перевірці знань.

1.4 Визначення ключових функцій та можливостей розроблюваного додатку

Проектування сучасного інструменту для створення вікторин передбачає визначення ключових функцій, які забезпечують гнучкість, інтерактивність і простоту у використанні. Основна ідея полягає у створенні конструктора, що дозволяє користувачам самостійно формувати структуру тесту, вибирати типи запитань та налаштовувати параметри проходження. Такий підхід відкриває можливості для широкого кола сценаріїв застосування – від навчання до розваг.

Особливістю розроблюваного додатку є використання штучного інтелекту для генерації контенту. Це дає змогу автоматизувати створення питань, адаптувати складність завдань та персоналізувати зміст відповідно до запиту користувача. Використання зовнішнього API, зокрема OpenAI, забезпечує доступ до потужних мовних моделей, які здатні формувати різноманітні формулювання запитань у реальному часі.

Для підвищення ефективності взаємодії з платформою реалізується функція створення кімнат, у яких вікторини можуть проходити одночасно декілька учасників. Передбачено систему підрахунку балів, формування результатів, а також можливість адміністрування процесу. Це дозволяє не лише проводити індивідуальні сесії тестування, а й організовувати змагання, тренінги або освітні заходи з елементами контролю й аналітики.

1.4.1 Функціонал створення вікторин через конструктор

Функціонал конструктора вікторин є центральною частиною розроблюваного додатку. Він надає можливість створювати запитання різних типів – з одним або кількома правильними варіантами, відкриті, зображеннями чи таймером. Інтерфейс конструктора побудований таким

чином, щоб навіть користувач без технічного досвіду міг легко сформувати вікторину відповідно до своїх потреб.

Користувач може додавати, редагувати та видаляти запитання, змінювати їхню послідовність, встановлювати вагу кожного завдання в загальній оцінці. Передбачена можливість попереднього перегляду вікторини перед її запуском, а також збереження проектів для подальшого редагування. Такий підхід забезпечує гнучкість і комфорт при створенні тестового контенту.

1.4.2 Використання OpenAI API для генерації питань

Інтеграція з OpenAI API дозволяє автоматизувати процес створення запитань, значно спрощуючи роботу організатора. Користувач може вказати тему або ключові слова, після чого система на основі штучного інтелекту генерує відповідні питання з варіантами відповідей. Такий підхід не лише економить час, а й забезпечує різноманітність формулювань, підвищуючи якість та інформативність вікторин.

1.4.3 Організація проходження вікторин через «кімнати»

Функція створення кімнат дозволяє організовувати структуровані сесії проходження вікторин, у яких беруть участь декілька користувачів одночасно. Це особливо корисно для проведення онлайн-занять, тренінгів або змагань у реальному часі. Кожна кімната має унікальний ідентифікатор, що забезпечує швидке підключення учасників без необхідності реєстрації або додаткових дій.

Організатор кімнати має змогу керувати процесом проходження: запускати вікторину, відслідковувати відповіді в реальному часі, контролювати таймінг і завершувати сесію. Це забезпечує інтерактивність і

підвищує рівень залучення учасників, які можуть бачити свій прогрес та, за потреби, конкурувати з іншими в рамках однієї групи.

Кімнати можуть бути відкритими або захищеними паролем, що дозволяє обмежити доступ лише для цільової аудиторії. Передбачена також можливість налаштування правил доступу – наприклад, за лінком або через запрошення. Такий функціонал робить систему гнучкою у використанні для різних форматів подій – від неформальних вікторин до офіційних освітніх сесій.

1.4.4 Система підрахунку балів та виведення результатів

Система підрахунку балів автоматично оцінює відповіді учасників відповідно до заздалегідь заданих правил, враховуючи правильність, швидкість виконання та інші параметри. Після завершення вікторини користувач отримує зведення результатів, яке може включати загальний бал, відсоток правильних відповідей, а також пояснення до кожного запитання. Це дозволяє швидко оцінити рівень знань і, за потреби, визначити напрямки для подальшого навчання.

1.4.5 Адміністрування процесу проходження та управління вікторинами

Адміністрування процесу проходження вікторин включає керування користувачами, контроль за активністю у кімнатах, модерацію контенту та доступ до розширеної аналітики.

Організатор може відстежувати динаміку відповідей, переглядати індивідуальні результати, блокувати порушників або переглядати статистику за групами. Такий рівень контролю дозволяє ефективно управляти як окремими сесіями, так і всією системою загалом.

1.5 Постановка задачі

Розробка конструктора вікторин як вебдодатку передбачає вирішення комплексу взаємопов’язаних задач, які охоплюють як етапи аналізу, так і проектування, реалізації та перевірки працездатності системи. Задача розробки полягає у створенні зручного, адаптивного і функціонального сервісу для створення та проходження вікторин з можливістю автоматичної генерації запитань на основі штучного інтелекту.

Для досягнення поставленої мети необхідно вирішити такі основні задачі: провести аналіз предметної області та огляд сучасних платформ для створення вікторин; виявити потреби цільової аудиторії та сформулювати функціональні та нефункціональні вимоги до системи; спроектувати архітектуру вебдодатку з урахуванням модульності, масштабованості та інтерактивності; реалізувати функціонал конструктора запитань, створення кімнат, підрахунку результатів та адміністрування сесій; інтегрувати OpenAI API для автоматичної генерації тестових запитань на основі введених тем або ключових слів; забезпечити зручний інтерфейс користувача з адаптивною верстkою для різних типів пристроїв; протестувати систему та оцінити її ефективність, стабільність і відповідність поставленим вимогам.

Вирішення зазначених задач дозволить створити повноцінну систему, здатну забезпечити сучасний користувацький досвід у сфері цифрового тестування, включаючи інтуїтивний інтерфейс, адаптивність до різних пристроїв та гнучкі механізми створення і проходження вікторин. Реалізація таких функцій, як генерація запитань на основі теми, миттєвий підрахунок балів, збереження прогресу та інтелектуальна перевірка відповідей, сприятиме не лише зручності користування, а й загальному підвищенню якості навчального процесу.

2 ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Аналіз та формалізація вимог

Етап аналізу та формалізації вимог є одним із ключових у процесі проектування програмного забезпечення. Його основне завдання полягає у виявленні, узагальненні та структуризації очікувань користувачів, а також технічних обмежень, у межах яких функціонуватиме система. Саме на цьому етапі формується уявлення про те, що має реалізовувати вебдодаток, як він буде взаємодіяти з користувачами та які характеристики повинні бути дотримані для забезпечення ефективності.

Вимоги до системи поділяються на функціональні та нефункціональні. Функціональні вимоги описують основні дії, які повинна виконувати система, зокрема створення вікторин, генерація запитань, проходження сесій, адміністрування результатів. Нефункціональні вимоги включають аспекти продуктивності, зручності використання, масштабованості, доступності та надійності. Урахування обох типів вимог дозволяє побудувати збалансовану систему, яка не лише виконує поставлені задачі, а й забезпечує стабільну роботу у реальному середовищі.

Окрему увагу приділено вимогам до безпеки, конфіденційності та захисту даних користувачів. Оскільки додаток передбачає збереження результатів вікторин, авторизацію через зовнішні сервіси та взаємодію з API, важливо забезпечити відповідність сучасним стандартам захисту інформації. Формалізація вимог на цьому етапі слугуватиме основою для подальшого проектування архітектури системи.

2.1.1 Функціональні вимоги

Основною функціональною вимогою до вебдодатку є можливість створення вікторин за допомогою зручного конструктора. Користувач

повинен мати змогу додавати різні типи запитань, вказувати варіанти відповідей, позначати правильні варіанти, а також налаштовувати параметри кожної вікторини. Конструктор має бути інтуїтивно зрозумілим, підтримувати перегляд і редагування збережених проектів.

Додаток повинен реалізовувати функцію генерації запитань на основі ключових слів або тем, які вводить користувач. Для цього передбачено інтеграцію з OpenAI API, що дозволяє автоматизувати створення текстового контенту. Користувач повинен отримувати одне або кілька згенерованих запитань із можливістю подальшого редагування перед збереженням у вікторину.

Необхідною функцією є організація проходження вікторини в режимі кімнати. Кожна вікторина має запускатися в окремій сесії, до якої можуть приєднуватися інші користувачі за посиланням. Кімната повинна мати унікальний ідентифікатор або код, що забезпечує приватність і контроль доступу. Організатор має змогу запускати, призупиняти та завершувати.

Система повинна забезпечувати підрахунок балів за результатами проходження вікторини. Результати мають формуватися автоматично на основі правильних відповідей, враховувати вагу запитань і кількість допущених помилок. Також необхідно реалізувати зведену статистику: кількість учасників, середній бал, розподіл відповідей.

Передбачено функціонал адміністрування вікторин. Організатор повинен мати змогу переглядати всі створені вікторини, редагувати їх, видаляти, дублювати або змінювати налаштування. Також має бути доступ до історії сесій, результатів окремих учасників та загальної аналітики по кожній вікторині.

Додатковими функціональними вимогами є система авторизації користувачів через Auth0, підтримка мультимовного інтерфейсу, адаптивність до різних пристроїв і можливість збереження прогресу у вікторинах.

2.1.2 Нефункціональні вимоги

Нефункціональні вимоги визначають характеристики системи, які не пов’язані безпосередньо з її функціональністю, але критично впливають на зручність, надійність і ефективність її використання. Одним з ключових аспектів є продуктивність вебдодатку. Система повинна забезпечувати швидку обробку запитів, мінімальні затримки при завантаженні інтерфейсів та стабільну роботу навіть при великій кількості одночасних користувачів.

Адаптивність інтерфейсу є ще однією важливою вимогою. Вебдодаток має коректно відображатися та функціонувати як на десктопних, так і на мобільних пристроях з різними розмірами екранів. Усі інтерактивні елементи повинні масштабуватися, зберігаючи доступність і зручність взаємодії незалежно від платформи. Це особливо важливо з огляду на широку цільову аудиторію, яка може користуватися системою в різних умовах.

Система повинна бути масштабованою, щоб забезпечити можливість її розширення у майбутньому без втрати стабільності. Це стосується як додавання нових функцій, так і збільшення обсягів оброблюваних даних або кількості одночасних підключень. Використання хмарної інфраструктури Google Cloud у поєднанні з Firebase дозволяє забезпечити горизонтальну масштабованість на рівні бекенду та бази даних.

Надійність системи полягає в забезпеченні стабільної роботи упродовж тривалого часу без збоїв і втрати даних. Для цього необхідно впровадити механізми резервного копіювання, обробки помилок та логування критичних подій. Крім того, важливо мінімізувати ймовірність виникнення помилок користувача за допомогою валідації форм і передбачуваної поведінки інтерфейсу.

Зручність використання або юзабіліті є ще одним критичним нефункціональним аспектом. Інтерфейс повинен бути інтуїтивним, з логічною структурою та зрозумілою навігацією. Реалізація через shadcn UI

та react-hook-form дозволяє створити простий і послідовний інтерфейс, який не потребує додаткових пояснень для користувача і сприяє швидкому освоєнню платформи.

Крім того, важливо забезпечити достатній рівень доступності. Додаток має враховувати потреби користувачів із різними можливостями, зокрема підтримувати використання через клавіатуру, мати контрастний дизайн, адаптуватися до програм екранного озвучування. Такий підхід не лише відповідає сучасним стандартам, а й розширює потенційну аудиторію системи.

2.1.3 Вимоги до безпеки та конфіденційності

Питання безпеки та конфіденційності є критично важливими для будь-якого вебдодатку, особливо якщо система працює з персональними даними користувачів або дозволяє автентифікацію через зовнішні сервіси. У розроблюваному додатку передбачається використання Auth0 для авторизації, що забезпечує стандартизований і безпечний механізм входу з підтримкою сучасних протоколів, таких як OAuth 2.0 і OpenID Connect.

Захист даних користувача повинен реалізовуватись на всіх рівнях системи. На рівні бази даних важливо обмежити доступ до конфіденційної інформації та впровадити контроль прав доступу на основі ролей. Firebase дозволяє реалізовувати правила доступу до колекцій у Firestore, що дає змогу жорстко регламентувати, хто і до яких документів має доступ у різних сценаріях взаємодії з системою.

Додаток повинен забезпечувати шифрування усіх переданих даних. Усі API-запити, включаючи автентифікацію, обмін даними з OpenAI, роботу з Firestore та Google Cloud Storage, мають проходити через захищене з'єднання HTTPS. Це дозволяє захистити дані від перехоплення та зменшити ризики атак типу man-in-the-middle.

Окрему увагу слід приділити захисту від типових вебуразливостей, таких як SQL-ін’єкції, XSS, CSRF та інші. Використання сучасних фреймворків, таких як Nest.js на серверній частині та React на клієнтській, дозволяє значно зменшити ці ризики завдяки вбудованим механізмам захисту. Важливо також реалізувати перевірку даних, що надходять від користувача, як на клієнті, так і на сервері.

Для підвищення надійності системи доцільно реалізувати логування критичних подій, таких як невдалі спроби входу, доступ до обмежених ресурсів або підозріла активність у кімнатах. Це дозволяє не лише своєчасно виявляти потенційні загрози, а й фіксувати факти порушення політик доступу для подальшого аналізу.

З метою відповідності чинному законодавству про захист персональних даних, система повинна включати політику конфіденційності, з якою користувач погоджується при першому вході. Крім того, необхідно забезпечити можливість видалення облікового запису та всіх пов’язаних із ним даних на запит користувача, що є однією з вимог сучасних стандартів приватності.

2.2 Проектування архітектури вебдодатку

Проектування архітектури вебдодатку є критично важливим етапом, оскільки саме архітектура визначає взаємодію компонентів, їхню відповідальність та стійкість до навантаження.

Архітектурна модель побудована відповідно до принципів багаторівневої клієнт-серверної взаємодії, де кожен рівень виконує окрему роль: представлення, обробка логіки та збереження даних.

Далі представлено загальну структурну схему вебдодатку з ключовими модулями (рисунок 2.1). Вона відображає основні компоненти системи та взаємозв’язки між ними.

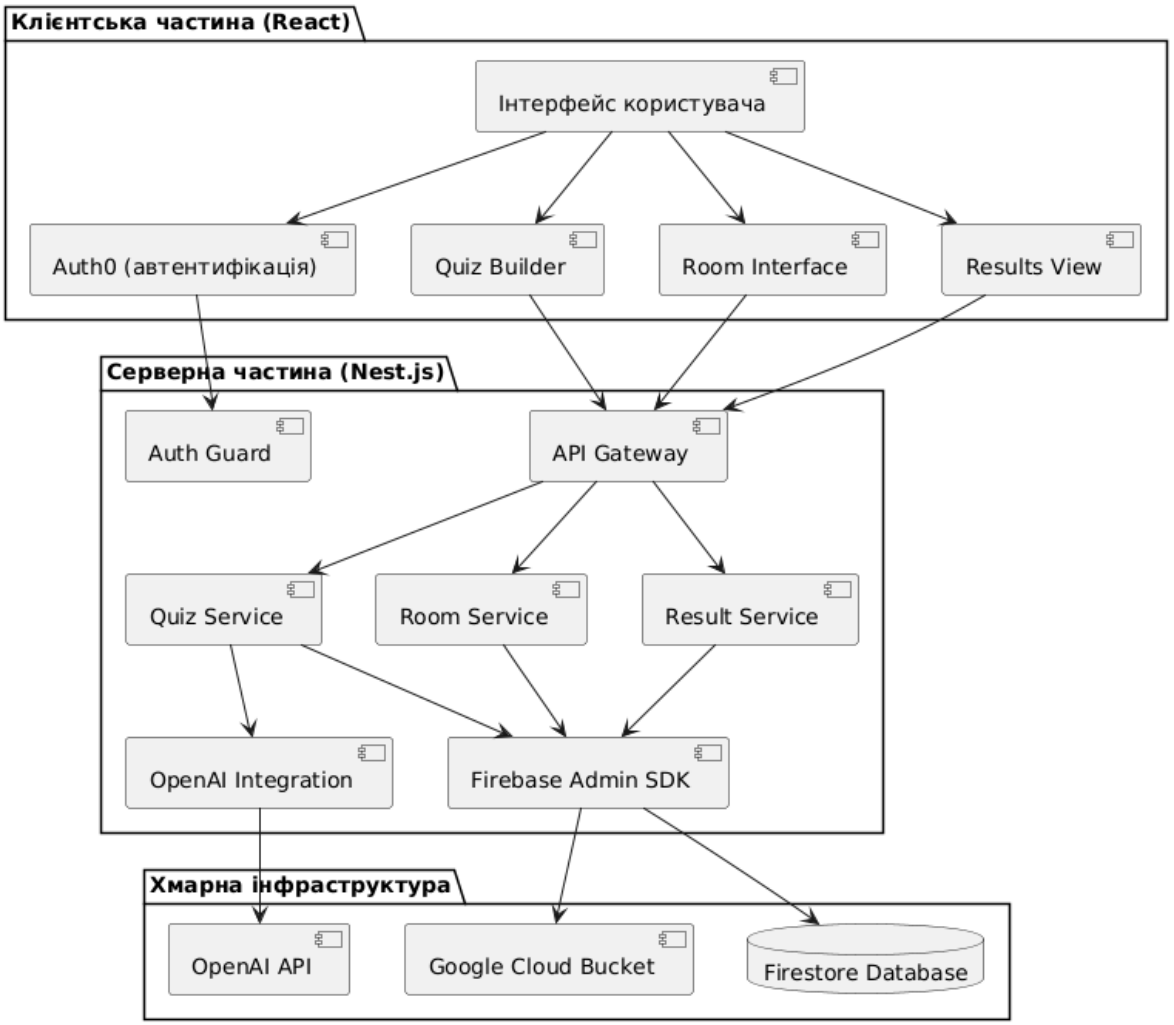

Рисунок 2.1 – Спроектована архітектура вебдодатку

На рівні клієнта реалізовано інтерфейс користувача з використанням бібліотеки React. Клієнт взаємодіє з такими модулями як Quiz Builder для створення вікторин, Room Interface для участі у сесії, Results View для перегляду результатів та Auth0 як інструмент автентифікації. Усі ці компоненти комунікують із серверною частиною через HTTP-запити, обробку яких забезпечує централізований API Gateway.

Серверна частина реалізована з використанням Nest.js, що дозволяє дотримуватися принципів модульності та масштабованості. Основними модулями є Quiz Service, Room Service і Result Service, кожен з яких відповідає за окрему групу операцій. Наприклад, Quiz Service обробляє запити на створення та редагування вікторин, Room Service керує запуском

і перебігом сесій, а Result Service відповідає за збереження та надання результатів проходження тестів.

Для автентифікації запитів сервер використовує компонент Auth Guard, який перевіряє авторизаційний токен, отриманий через Auth0. Така перевірка є необхідною для забезпечення доступу лише до дозволених ресурсів. Додатково реалізовано інтеграцію з OpenAI API, що використовується для автоматичної генерації запитань у процесі створення вікторини. Ця інтеграція здійснюється виключно через відповідний внутрішній сервіс, що спрощує масштабування та контроль.

Обробка даних та їх збереження реалізовані на основі Firebase Admin SDK, який взаємодіє з базою Firestore Database та Google Cloud Bucket. Firestore зберігає структуровані колекції документів, зокрема дані про вікторини, користувачів, сесії та відповіді. У випадках, коли потрібне збереження файлів або зображень, використовується Google Cloud Bucket як хмарне файлове сховище.

Подібна модульна архітектура дозволяє легко розширювати систему новими функціями, масштабувати окремі компоненти залежно від навантаження, а також реалізовувати гнучке адміністрування прав доступу. Така структура також спрощує тестування й оновлення окремих частин додатку без ризику порушення загальної стабільності системи.

2.3 Вибір технологій для створення конструктору

Під час розробки вебдодатку для конструктора вікторин було прийнято рішення використовувати сучасний і перевірений стек технологій, який забезпечує надійність, гнучкість та швидкість розробки. На клієнтській стороні обрано бібліотеку React, що дозволяє реалізувати адаптивний і компонентний інтерфейс із високим рівнем взаємодії користувача. Додатково використовується Vite як інструмент для швидкої збірки і розгортання, що пришвидшує цикл розробки.

Інтерфейс реалізовано з використанням UI-бібліотеки shadcn UI, яка базується на Tailwind CSS і забезпечує чистий, сучасний дизайн компонентів. Для управління маршрутизацією в додатку застосовується react-router, що дозволяє реалізувати логічну навігацію між екранами: конструктором, сесією проходження, результатами тощо. Для роботи з формами використано react-hook-form – легкий інструмент для обробки валідацій і взаємодії з формами у React, що дозволяє зменшити кількість коду і підвищити якість валідації на фронтенді.

Для реалізації автентифікації обрано сервіс Auth0, який підтримує вхід через сторонні облікові записи, безпечне управління сесіями, оновлення токенів і відповідає сучасним вимогам до безпеки. Це дозволяє уникнути ручної реалізації механізмів логіну, обробки паролів та зберігання сесій. Auth0 легко інтегрується з React та Nest.js, що дозволяє забезпечити узгоджений підхід до ідентифікації на всіх рівнях додатку.

На серверній частині застосовується Nest.js – потужний фреймворк для створення масштабованих серверних застосунків на основі Node.js. Nest.js базується на TypeScript та підтримує модульну архітектуру, що значно полегшує підтримку коду. У межах цього проекту Nest.js використовується для створення API, обробки бізнес-логіки, інтеграції з Firebase та сторонніми сервісами, такими як OpenAI.

Зберігання даних реалізовано на основі Firestore – масштабованої хмарної бази даних від Google, яка ідеально підходить для роботи в режимі реального часу. Firestore забезпечує зберігання структурованих документів у колекціях, дозволяє здійснювати гнучкі запити та застосовувати правила безпеки для контролю доступу. Завдяки використанню Firebase SDK доступ до бази можна реалізувати як з клієнтської, так і з серверної частини, з дотриманням відповідних політик.

Для зберігання великих файлів, таких як зображення, передбачено використання Google Cloud Storage (Bucket), що інтегрується через Firebase Admin SDK. Усі компоненти додатку запускаються в контейнерах Docker і

деплояться у Google Cloud, що забезпечує високу надійність, автоматичне масштабування та можливість централізованого моніторингу. Такий вибір технологій дозволяє розробити стабільний, безпечний і сучасний додаток з можливістю подальшого розширення.

2.4 Проектування бази даних

База даних є ключовим компонентом у вебдодатку конструктора вікторин, оскільки зберігає інформацію про користувачів, створені вікторини, запитання, відповіді, сесії проходження та результати. Для реалізації обрано Firestore – документоорієнтовану хмарну базу даних від Google, що надає можливість гнучкої організації структури з високою продуктивністю та підтримкою масштабування. Firestore працює з колекціями та документами, що дозволяє зберігати кожну вікторину або сесію як окрему сутність зі своїми вкладеними атрибутами.

Основними колекціями в базі даних є Users, Quizzes, Sessions та Results. Колекція Users містить документи з інформацією про кожного користувача: його ідентифікатор, ім’я, дату реєстрації, тип доступу та інші метадані. Колекція Quizzes зберігає структуру кожної вікторини, зокрема назву, опис, набір запитань, типи відповідей та логіку оцінювання. Запитання можуть зберігатися у вигляді вкладених масивів або як підколекція залежно від складності структури вікторини.

Колекція Sessions використовується для фіксації активних або завершених сесій проходження вікторин. Кожна сесія містить посилання на конкретну вікторину, список учасників, статус сесії (активна, завершена, скасована) та часові мітки. Це дозволяє реалізовувати паралельне проходження однієї вікторини кількома користувачами, зберігаючи ізольовані результати для кожної групи. Додатково можна зберігати параметри сесії, такі як тривалість, тип заліку або додаткові налаштування.

Результати проходження вікторин фіксуються у колекції Results. Вона включає детальну інформацію про кожну відповідь користувача, кількість набраних балів, правильність відповідей та аналітику на основі результатів сесії. Така структура дозволяє реалізувати перегляд особистих результатів, а також статистику по групі учасників. Залежно від потреб, результати можуть зберігатися як підколекція в Sessions або у вигляді окремої колекції з посиланнями на сесію та користувача.

З міркувань безпеки й продуктивності у Firestore застосовуються правила доступу, які обмежують читання та запис даних. Наприклад, користувачі можуть мати доступ лише до своїх вікторин або результатів, а організатор – до статистики лише своїх сесій. Це дозволяє уникнути витоку конфіденційної інформації та забезпечити розмежування прав у межах додатку.

Firestore також підтримує індексацію полів, що дає змогу швидко виконувати запити, наприклад за ідентифікатором користувача, датою створення, назвою вікторини або статусом сесії. Завдяки цьому база даних залишається ефективною навіть при великій кількості одночасних користувачів і запитів. Гнучкість структури Firestore і можливість автоматичного масштабування роблять її оптимальним вибором для потреб цього вебдодатку.

2.5 Сценарії взаємодії та логіка користувацьких потоків

Проектування сценаріїв взаємодії дозволяє формалізувати поведінку користувача під час роботи з системою та описати послідовність обміну повідомленнями між компонентами додатку. У цьому контексті важливою складовою є побудова діаграм послідовності, які відображають динаміку взаємодії між користувачем, клієнтським інтерфейсом, серверною частиною та зовнішніми сервісами. Такий підхід дає змогу не лише краще

зрозуміти логіку роботи системи, а й полегшити процес реалізації окремих функціональних блоків.

У контексті конструктора вікторин виділяються дві основні ролі: організатор та учасник. Кожна з них має власний набір сценаріїв взаємодії з додатком. Організатор відповідає за створення контенту, запуск вікторин і адміністрування сесій, тоді як учасник бере участь у проходженні тестування та отримує результати. Далі наведено діаграми послідовності, які відображають логіку дій для кожної з ролей у типовій взаємодії з системою. Сценарій взаємодії для ролі організатора (рисунок 2.2) описує ключові етапи роботи з вебдодатком – від автентифікації до запуску сесії вікторини.

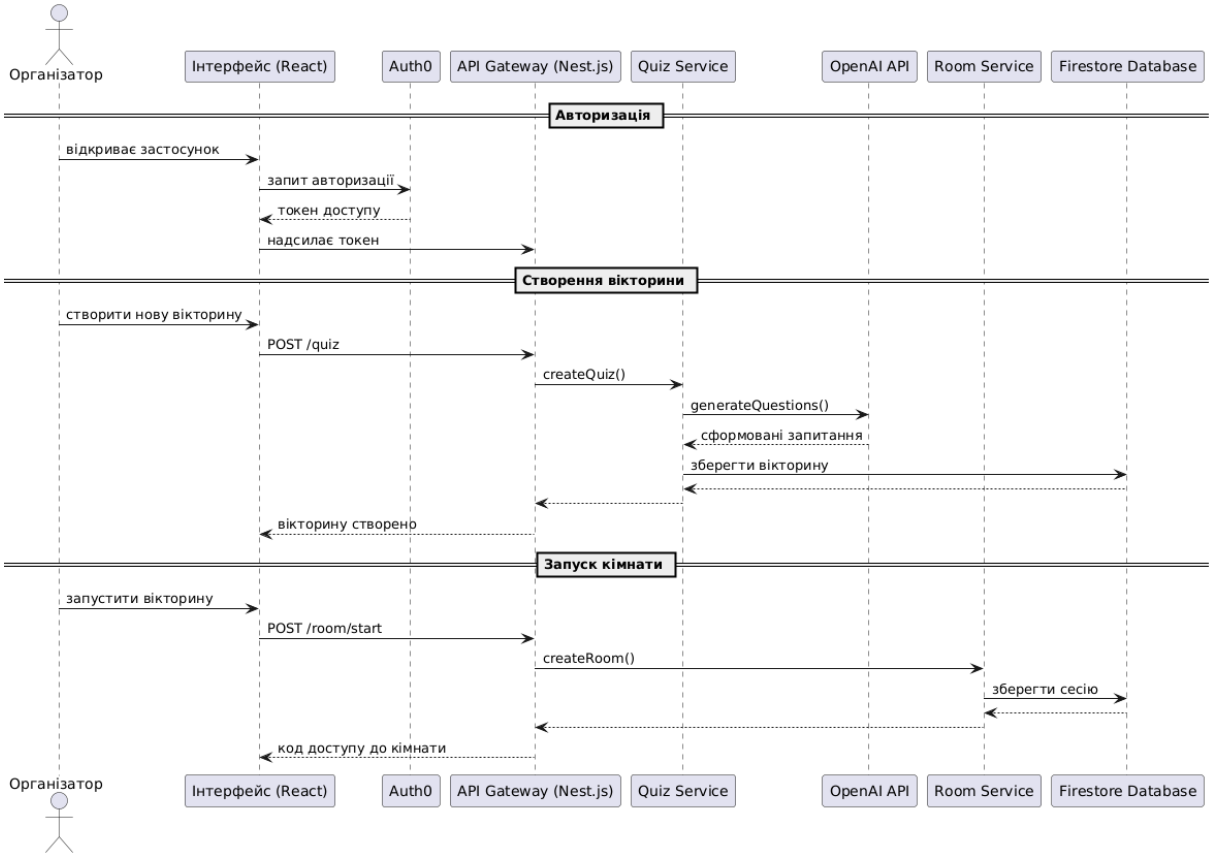

Рисунок 2.2 – Сценарій взаємодії для ролі організатора

Вище можна побачити діаграму послідовності, яка ілюструє основні дії організатора в системі, а також їхню послідовність та взаємодію з внутрішніми і зовнішніми сервісами. Це дає змогу наочно простежити логіку обміну повідомленнями між компонентами системи.

Після відкриття застосунку організатор ініціює процес авторизації через сервіс Auth0, який повертає токен доступу. Токен використовується для автентифікації всіх подальших запитів на сервер. Після входу в систему користувач має змогу створити нову вікторину, використовуючи відповідний інтерфейс конструктора. З формою створення вікторини пов’язано надсилання запиту до серверної частини, яка викликає внутрішній сервіс генерації запитань через інтеграцію з OpenAI API.

Згенеровані запитання повертаються на сервер, після чого вся структура вікторини зберігається у базі даних Firestore. У відповідь користувач отримує підтвердження про успішне створення вікторини. Далі організатор може запустити сесію вікторини для учасників, надіславши запит до Room Service. Цей сервіс створює нову кімнату, яка прив’язана до обраної вікторини, зберігає сесію в базі даних і повертає код доступу.

Організатор отримує унікальний код кімнати, який надалі надсилається учасникам для приєднання до вікторини. Дана логіка дозволяє запускати вікторини як для приватних груп, так і для відкритих аудиторій. Уся взаємодія з системою відбувається послідовно та охоплює ключові компоненти: інтерфейс користувача, API Gateway, спеціалізовані сервіси та зовнішні платформи, включно з OpenAI і Firestore.

Сценарій взаємодії учасника охоплює процес приєднання до вже запущеної сесії вікторини, проходження запитань та отримання результату. Далі зображено діаграму послідовності, яка ілюструє основні дії учасника та взаємодію з ключовими компонентами системи, включно з Room Service, Result Service та Firestore Database (рисунок 2.3). Така візуалізація дає змогу зрозуміти динаміку обміну повідомленнями на кожному етапі.

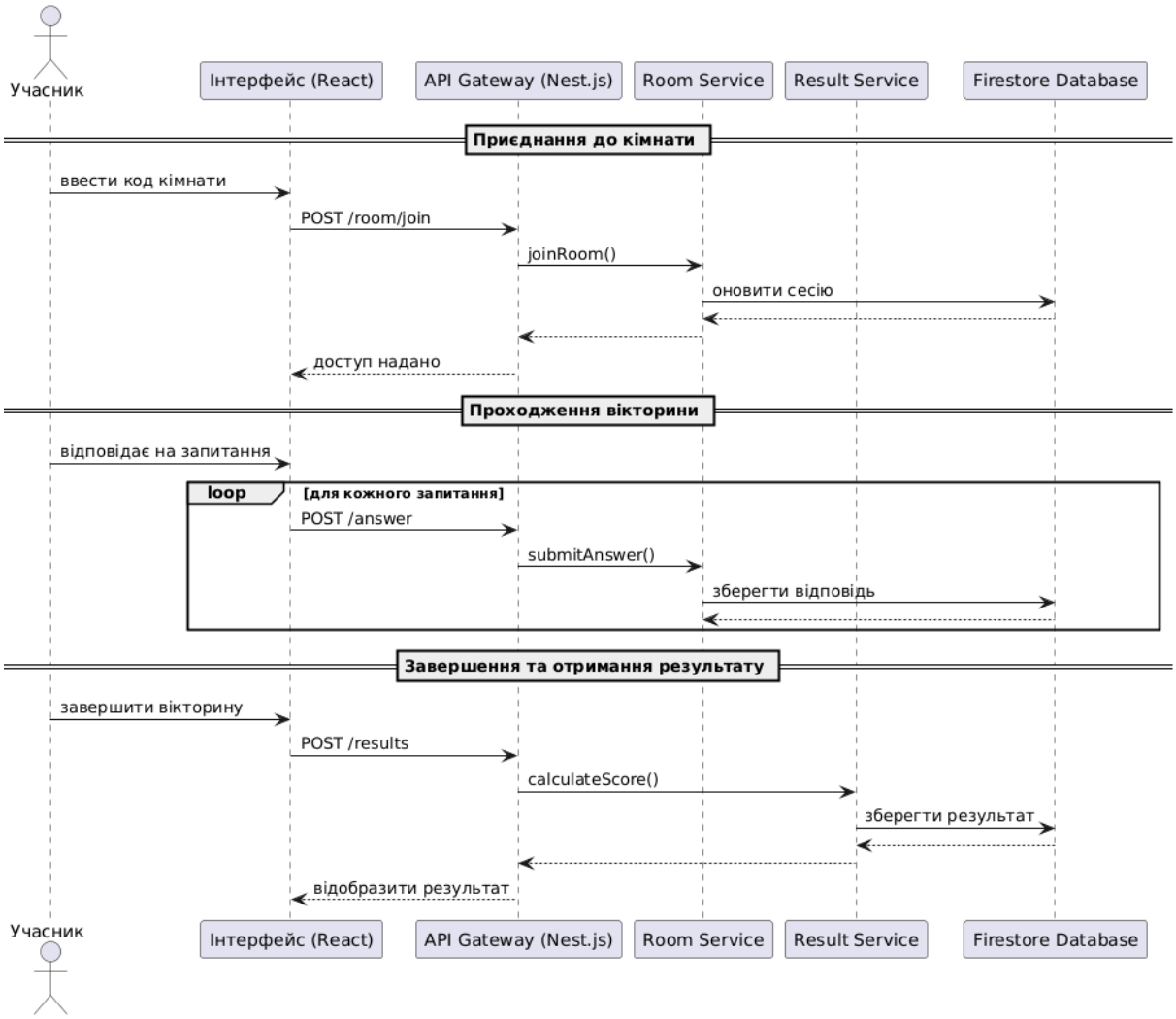

Рисунок 2.3 – Сценарій взаємодії для ролі учасника

Учасник вводить код кімнати, після чого відбувається перевірка на сервері та оновлення даних сесії. Якщо доступ надано, користувач може відповідати на запитання, що надходять у циклі. Кожна відповідь зберігається в базі даних через Room Service. Після завершення проходження надсилається запит на підрахунок результату, який обробляється Result Service. Система формує підсумок і зберігає результат у Firestore, після чого він відображається учаснику. Увесь процес є послідовним та інтерактивним, що забезпечує ефективну взаємодію і миттєвий зворотний зв’язок.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Використані інструменти та технології

У процесі реалізації вебдодатку конструктора вікторин було використано сучасний стек технологій, який охоплює клієнтську, серверну, базову та інфраструктурну частини системи. Підбір інструментів здійснювався з урахуванням таких факторів, як швидкість розробки, можливість масштабування, підтримка стандартів безпеки, інтеграція з зовнішніми API та зручність супроводу проекту. Основна мета вибору технологій полягала в забезпеченні стабільної роботи вебдодатку, інтуїтивного інтерфейсу для користувачів, легкого розширення функціоналу в майбутньому, а також використання готових рішень для типових задач, зокрема автентифікації та генерації контенту на основі штучного інтелекту.

Клієнтську частину вебдодатку реалізовано за допомогою бібліотеки React – одного з найпопулярніших інструментів для створення інтерфейсів користувача. React дозволяє розробляти компонентну структуру, де кожен елемент інтерфейсу описується окремо, що значно спрощує його підтримку та повторне використання. Основною перевагою React є віртуальний DOM, який оптимізує оновлення сторінки, що особливо важливо для динамічних частин застосунку, таких як конструктор запитань або кімната вікторини. Крім того, React підтримується великою спільнотою, має потужну документацію та широке коло сумісних бібліотек, що було важливим критерієм при виборі технології.

Для управління навігацією в межах клієнтського інтерфейсу використано бібліотеку react-router. Вона дозволяє реалізовувати SPA, single page application – тобто односторінковий вебдодаток, у якому користувач взаємодіє з різними сторінками без повного перезавантаження. У межах цього проекту react-router відповідає за перемикання між екраном створення

вікторини, участі у сесії, відображення результатів тощо. Це підвищує швидкість взаємодії та загальний користувацький досвід.

Для обробки форм і валідації введених даних обрано бібліотеку react-hook-form. Вона забезпечує ефективну роботу з формами, дозволяє мінімізувати ререндеринг компонентів, підтримує вбудовану й кастомну валідацію та має простий API. У вебдодатку ця бібліотека використовується як у формі створення вікторини, так і при введенні коду кімнати або заповненні відповідей. Поєднання з TypeScript дозволяє забезпечити типобезпечність і зменшити кількість помилок під час компіляції.

Для стилізації інтерфейсу обрано UI-бібліотеку shadcn UI, яка побудована на основі Tailwind CSS. Вона забезпечує набір готових адаптивних компонентів з сучасним дизайном, підтримує теми, а також гнучке налаштування зовнішнього вигляду без написання CSS-коду. Завдяки використанню shadcn UI вдалося реалізувати інтерфейс, який добре виглядає як на десктопних, так і на мобільних пристроях, зберігаючи при цьому високу швидкість завантаження.

Клієнтська частина компілюється за допомогою Vite – сучасного білдера, який значно пришвидшує процес розробки порівняно з класичними інструментами, такими як Webpack. Vite забезпечує миттєвий запуск проекту, гаряче оновлення модулів під час розробки та ефективну оптимізацію фінальної збірки. Завдяки цьому скорочено час розробки, що особливо важливо у кваліфікаційній роботі з обмеженими термінами реалізації.

Серверну частину вебдодатку реалізовано з використанням фреймворку Nest.js – сучасного бекенд-рішення, побудованого на Node.js і TypeScript. Nest.js дотримується принципів модульності, інверсії залежностей та декораторів, що дозволяє будувати зрозумілу та розширювану архітектуру. У проекті Nest.js використовується для створення API, обробки логіки взаємодії з базою даних, викликів до OpenAI API, управління сесіями та обліку результатів.

Зберігання структурованих даних реалізовано за допомогою сучасного Firestore – хмарної бази даних від Google, яка працює у форматі колекцій та документів. Firestore забезпечує високу швидкодію, автоматичне масштабування, підтримку запитів у реальному часі, а також вбудовану систему правил безпеки. У межах цього вебдодатку в Firestore зберігаються колекції користувачів, вікторин, сесій і результатів. Бібліотека Firebase Admin SDK використовується для взаємодії з Firestore із серверної частини Nest.js.

Для зберігання медіафайлів та інших об’єктів застосовується Google Cloud Storage (Bucket). Цей сервіс інтегрується з Firebase і дозволяє надійно зберігати файли, що можуть використовуватись у вікторинах, наприклад зображення до запитань. Bucket підтримує різні рівні доступу, керування метаданими та швидке отримання об’єктів за URL-посиланнями.

Автентифікація користувачів реалізована через сервіс Auth0, який підтримує широкий перелік стратегій входу – зокрема через електронну пошту, соціальні мережі або корпоративні облікові записи. Auth0 забезпечує генерацію захищених токенів, які використовуються на фронтенді та в API-запитах для контролю доступу до ресурсів. Крім того, інтеграція з Auth0 дозволяє централізовано управляти сесіями, автентифікацією та правами доступу без потреби розробляти ці функції вручну.

Однією з ключових особливостей вебдодатку є автоматична генерація запитань для вікторини. Цей функціонал реалізовано шляхом інтеграції з OpenAI API. Користувач має змогу задати тему або ключові слова, а система на їх основі формує змістовні тестові запитання. Для взаємодії з OpenAI API використовується HTTP-запит з авторизаційним ключем. Отриманий результат обробляється на сервері й передається клієнту у зручному форматі.

Розгортання проекту здійснюється в хмарному середовищі Google Cloud за допомогою Docker. Усі компоненти вебдодатку упаковано в окремі контейнери, що дозволяє зручно розгортати застосунок на будь-якій

інфраструктурі, яка підтримує Docker. Такий підхід забезпечує незалежність середовища, простоту масштабування, можливість оновлення без простоїв і централізований контроль за процесом виконання.

Таким чином, комбінація обраних інструментів і технологій дозволяє створити сучасний, масштабований та надійний вебдодаток, який поєднує в собі зручний інтерфейс, гнучку логіку обробки даних, інтеграцію зі штучним інтелектом та повноцінне хмарне середовище розгортання. Кожен із використаних компонентів виконує чітко визначену роль у загальній архітектурі системи, забезпечуючи її цілісність, продуктивність і готовність до реального застосування.

3.2 Реалізація клієнтської частини

У клієнтській частині застосунку ключову роль відіграє конфігурація інтерфейсу до бекенду через бібліотеку Redux Toolkit Query, яка дозволяє спростити виконання запитів та керування кешем. У лістингу 3.1 подано код, який визначає базовий API-об'єкт з іменем mainApi. Він використовує функцію createApi для створення джерела даних, яке охоплює всі необхідні запити до серверної частини. У параметрах конфігурації вказано базову URL-адресу, отриману з середовищної змінної, а також механізм підготовки заголовків запиту з авторизаційним токеном користувача.

У розділі endpoints цього ж коду, поданого у лістингу 3.1, визначено набір запитів і мутацій для роботи з сутностями користувачів, вікторин та кімнат. Наприклад, використано getCurrentUser для отримання даних поточного користувача, createQuiz і createRoom для створення нової вікторини або кімнати відповідно, а також submitAnswers і generateAnswer для надсилання відповідей та генерації додаткових варіантів через інтеграцію з OpenAI. Кожен запит пов’язано з відповідними тегами кешування, що забезпечує автоматичне оновлення даних.

Лістинг 3.1 – Програмний код, що реалізує клієнтську логіку взаємодії з бекендом

```
export const api = createApi({
  reducerPath: "mainApi",
  tagTypes: ["CurrentUser", "Quiz", "Room"],
  baseQuery: fetchBaseQuery({
    baseUrl: env.VITE_API_URL,
    prepareHeaders: (headers, { getState }) => {
      const token = (getState() as RootState).auth.accessToken
      if (token) headers.set("authorization", `Bearer ${token}`)
      return headers},}),
  endpoints: (builder) => ({
    getCurrentUser: builder.query({ query: () => "/users/current", providesTags: ["CurrentUser"] }),
    createQuiz: builder.mutation({ query: (body) => ({ url: "/quizzes", method: "POST", body }), invalidatesTags: ["CurrentUser"] }),
    getQuiz: builder.query({ query: (id) => `/quizzes/${id}`, providesTags: ["Quiz"] }),
    getRoom: builder.query({ query: (id) => `/rooms/${id}`, providesTags: ["Room"] }),
    createRoom: builder.mutation({ query: ({ quizId, name }) => ({ url: "/rooms", method: "POST", body: { quizId, name } }), invalidatesTags: ["Room", "Quiz", "CurrentUser"] }),
    submitAnswers: builder.mutation({ query: ({ roomId, ...body }) => ({ url: `/rooms/${roomId}/submit-answers`, method: "POST", body }), invalidatesTags: ["Room"] }),
    generateAnswer: builder.mutation({ query: ({ question, answers }) => ({ url: "/quizzes/generate-answer", method: "POST", body: { question, answers } }) }),
  })})
```

У лістингу 3.2 наведено програмний код, що відповідає за рендеринг головної сторінки вебдодатку з вітальним заголовком та кнопкою входу. Анімація реалізована за допомогою бібліотеки Framer Motion, а авторизація здійснюється через Auth0.

Лістинг 3.2 – Програмний код, що реалізує головну сторінку вебдодатку з анімованим заголовком і кнопкою входу

```
export const HomePage: FC = () => {
  const location = useLocation()
  const from = location.state?.from
  const { loginWithRedirect } = useAuth0()
  return (
    <HeroHighlight>
      <motion.h1 initial={{ opacity: 0, y: 20 }} animate={{ opacity: 1, y: [20, -5, 0] }} transition={{ duration: 0.5, ease: [0.4, 0.0, 0.2, 1] }} className="text-2xl px-4 md:text-4xl lg:text-5xl font-bold text-neutral-700 dark:text-white max-w-4xl leading-relaxed lg:leading-snug text-center mx-auto">
        Challenge Your Friends: The Ultimate Quiz Showdown!
      </motion.h1>
      <motion.div initial={{ opacity: 0, y: 20 }} animate={{ opacity: 1, y: [20, -5, 0] }} transition={{ delay: 1, duration: 0.5, ease: [0.4, 0.0, 0.2, 1] }} className="flex items-center justify-center mt-5">
    <Button onClick={() => loginWithRedirect({ appState: { returnTo: from ?? location.pathname } })}>Log in</Button>
    </HeroHighlight>)}
```

У лістингу 3.3 подано реалізацію сторінки, яка відображає інформаційну панель користувача зі списком створених ним вікторин. Також реалізовано кнопку для переходу до сторінки створення нової вікторини.

Лістинг 3.3 – Програмний код, що відображає дашборд користувача зі списком створених вікторин та кнопкою для переходу до створення нової

```
export const QuizzesListPage: FC = () => {
  const { data, isLoading, isError } = useGetCurrentUserQuery()
  const navigate = useNavigate()
  if (isLoading) return <LoadingOverlay />
  if (!data || isError) return <>Something went wrong</>
  return (
    <div className="container mx-auto px-4 py-8">
      <div className="flex items-center justify-between">
        <h1 className="text-3xl font-bold mb-8">Quiz Dashboard</h1>
        <div className="mb-8">
          <Button onClick={() => navigate("/quizzes/create")}>
            <PlusCircle className="mr-2 h-4 w-4" />
            Create new quiz
          </Button>
        </div>
      </div>
<div className="grid gap-6 md:grid-cols-2 lg:grid-cols-3">
{data.quizzes.map(quiz => <QuizCard key={quiz.id} quiz={quiz} />)}</div></div>)}
```

У лістингу 3.4 представлено програмний код, що реалізує логіку створення нової вікторини через клієнтський інтерфейс. Користувач послідовно додає запитання різних типів, вибираючи їх із попередньо визначеного переліку. Для кожного запитання передбачена окрема форма введення залежно від його типу, а після завершення останнього кроку відкривається модальне вікно з підсумком. Усі запитання зберігаються у локальному стані та оновлюються відповідно до дій користувача.

Лістинг 3.4 – Програмний код, що реалізує сторінку створення вікторини

```
export const CreateQuizPage: FC = () => {
  const [questions, setQuestions] = useState<Question[]>([])
  const [currentQuestionIndex, setCurrentQuestionIndex] = useState(0)
  const [isCreateModalOpen, setIsCreateModalOpen] = useState(false)
  const createQuestion = (type: QuestionType) => {
    setQuestions(prev => [...prev, createEmptyQuestion(type)])}
  const addQuestion = (data: NewQuestionData) => {
    const updated = updateQuestion(data)
    setQuestions(prev => [...prev.slice(0, -1), updated])
    setCurrentQuestionIndex(prev => prev + 1)}
  const handleFinish = (data: NewQuestionData) => {
    const updated = updateQuestion(data)
    setQuestions(prev => [...prev.slice(0, -1), updated])
    setIsCreateModalOpen(true)}
  const returnBack = () => {
    setCurrentQuestionIndex(prev => Math.max(prev - 1, 0))
    setQuestions(prev => prev.slice(0, -1))}
  return ( <>
     <div className="container mx-auto px-4 py-8 h-screen">
        {currentQuestionIndex === questions.length
          ? <HoverEffect items={quizTypes.map(i => ({ ...i, onClick: () => createQuestion(i.type) }))} className="h-full" />
          : <CreateQuestion question={questions[currentQuestionIndex]} onBack={returnBack} onFinish={handleFinish} onNewQuestion={addQuestion} />}</div>
      <CreateQuizModal open={isCreateModalOpen} onOpenChange={setIsCreateModalOpen} questions={questions} />)}
const CreateQuestion: FC<{ question: Question, onBack: () => void, onFinish: (d: NewQuestionData) => void, onNewQuestion:
```

Продовження лістингу 3.4

```
(d: NewQuestionData) => void }> = ({ question, onBack, onFinish, onNewQuestion }) => {
  switch (question.type) {
    case "single": return <CreateSingleQuestion question={question} onBack={onBack} onFinish={d => onFinish({ ...d, type: question.type })} onNewQuestion={d => onNewQuestion({ ...d, type: question.type })} />
    case "multiple": return <CreateMultipleQuestion question={question} onBack={onBack} onFinish={d => onFinish({ ...d, type: question.type })} onNewQuestion={d => onNewQuestion({ ...d, type: question.type })} />
    case "custom": return <CreateCustomAnswerQuestion question={question} onBack={onBack} onFinish={d => onFinish({ ...d, type: question.type })} onNewQuestion={d => onNewQuestion({ ...d, type: question.type })} />
    case "ai": return <CreateAIPoweredQuestion question={question} onBack={onBack} onFinish={d => onFinish({ ...d, type: question.type })} onNewQuestion={d => onNewQuestion({ ...d, type: question.type })} />}}
```

У лістингу 3.5 наведено реалізацію сторінки перегляду вікторини, яка завантажується за ідентифікатором з URL. Компонент відображає назву вікторини, її опис, кількість запитань, а також список відкритих кімнат, у яких можна пройти цю вікторину. Додатково передбачена можливість створення нової кімнати та виходу з облікового запису.

Лістинг 3.5 – Програмний код, що відображає сторінку вікторини з її описом, кількістю запитань, списком активних кімнат

```
export const QuizPage: FC = () => {
  const { quizId } = useParams<{ quizId: string }>()
  const { data, isLoading, isError } = useGetQuizQuery(quizId!)
  const { logout } = useAuth0()
```

Продовження лістингу 3.5

```
      const [isCreateRoomModalOpen, setIsCreateRoomModalOpen]
= useState(false)
      if (isLoading) return <LoadingOverlay />
      if (isError || !data) return <div>Error</div> return (<>
          <div className="container mx-auto px-4 py-8">
            <Card className="mb-8">
              <div className="flex items-center gap-5">
                <CardHeader>
    <CardTitle className="text-3xl">{data.name}</CardTitle>
    <CardDescription className="flex items-center text-base">
                    <div className="flex items-center gap-2">
                      <Users className="h-5 w-5" />
                      {data.questions.length}
{data.questions.length === 1 ? "question" : "questions"}
                    </div>
                  </CardDescription>
                </CardHeader>
                <Button             onClick={()             =>
setIsCreateRoomModalOpen(true)}>Create room</Button>
              </div>
              <CardContent>
                <p      className="text-base      text-muted-
foreground">{data.description}</p>
              </CardContent>
            </Card>
    <h2 className="text-2xl font-semibold mb-4">Open Rooms</h2>
            <RoomList rooms={data.rooms} />
            <Button  className="mt-8"  onClick={()  =>  logout({
logoutParams:    {    returnTo:    window.location.origin    }
})}>Logout</Button></div>
          <CreateRoomModal                    quizId={quizId!}
open={isCreateRoomModalOpen}
onOpenChange={setIsCreateRoomModalOpen} /></>)}
```

У лістингу 3.6 подано реалізацію компонента, що відповідає за проходження вікторини користувачем у межах певної кімнати. Користувач спочатку вводить своє ім’я, після чого поетапно відповідає на запитання різних типів. Усі відповіді накопичуються та після завершення надсилаються на сервер для перевірки, включно з обробкою AI-питань. По завершенні проходження відображається таблиця з результатами усіх учасників цієї кімнати.

Лістинг 3.6 – Програмний код, що реалізує проходження вікторини користувачем

```
export const TakeQuiz: FC<TakeQuizProps> = ({ room }) => {
  const quiz = room.quiz
  const [name, setName] = useState("")
  const [currentQuestionIndex, setCurrentQuestionIndex] = useState(0)
  const [answers, setAnswers] = useState<any[]>([])
  const [submitAnswers, { isLoading }] = useSubmitAnswersMutation()
  const currentQuestion = quiz.questions[currentQuestionIndex]
  const handleAnswer = (answer: any) => {
    const newAnswers = updateItemAtIndex(answers, currentQuestionIndex, answer)
    if (currentQuestionIndex +1< quiz.questions.length) {
      setAnswers(newAnswers)
      setCurrentQuestionIndex(i => i + 1)
      return}
    const apiAnswers = newAnswers.map((a, i) => {
      const q = quiz.questions[i]
      if (q.type === "single") return { type: q.type, answerId: a, questionId: q.id }
      if (q.type === "multiple") return { type: q.type, answerIds: a, questionId: q.id }
      return { type: q.type, answer: a, questionId: q.id }}
```

Продовження лістингу 3.6

```
        submitAnswers({ answers: apiAnswers, name, roomId:
room.id }).unwrap().then(() => {
          setAnswers(newAnswers)
          setCurrentQuestionIndex(i => i + 1)})}
      const renderQuestion = (question: Question) => {
        if (question.type === "single") return
<SingleAnswerQuestionForm isLoading={isLoading}
question={question} onAnswer={handleAnswer} />
        if (question.type === "multiple") return
<MultipleAnswerQuestionForm isLoading={isLoading}
question={question} onAnswer={handleAnswer} />
        if (question.type === "custom") return
<CustomAnswerQuestionForm isLoading={isLoading}
question={question} onAnswer={handleAnswer} />
        if (question.type === "ai") return
<AIPoweredQuestionForm isLoading={isLoading}
question={question} onAnswer={handleAnswer} />
        return null}
    return ( <div className="h-screen w-full flex items-center
justify-center relative">)
```

Реалізація клієнтської частини вебдодатку забезпечує повноцінну взаємодію користувача з системою створення та проходження вікторин. Інтерфейс реалізовано з використанням сучасного стеку технологій, що включає React, TypeScript, бібліотеки Tailwind та Framer Motion для візуальних ефектів, а також Redux Toolkit Query для зручної роботи з API.

Завдяки чіткому поділу на компоненти вдалося реалізувати масштабовану архітектуру, що охоплює створення вікторин, управління кімнатами та інтерактивне проходження питань. Користувацький досвід доповнено підтримкою різних типів запитань, включно з інтеграцією штучного інтелекту для динамічного формування та перевірки відповідей.

3.3 Реалізація серверної частини

У лістингу 3.7 представлено програмний код, що виконує створення нового користувача в базі даних Firestore на основі ідентифікатора Auth0. Під час цього також завантажується аватар користувача у хмарне сховище та зберігається відповідне посилання.

Лістинг 3.7 – Програмний код, що створює нового користувача

```
@Injectable()
export class AuthService {
  constructor(private firebaseService: FirebaseService) {}
  async createAuth0User({ auth0Id, email, name, avatar }: CreateAuth0User): Promise<void> {
    const db = this.firebaseService.getFirestore();
    const userRef = db.collection('users').doc();
    const avatarData = await fetch(avatar).then(res => {
      if (!res.ok) throw new Error('Failed to fetch avatar');
      return res.arrayBuffer();});
    const bucket = getStorage().bucket();
    const fileName = `avatars/${Buffer.from(email).toString('base64')}.jpg`;
    const file = bucket.file(fileName);
  await file.save(Buffer.from(avatarData), { public: true });
    const fileUrl = `https://storage.googleapis.com/${bucket.name}/${fileName}`;
    await firestore().runTransaction(async tx => {
      const existing = await tx.get(db.collection('users').where('auth0Id', '==', auth0Id));
      if (!existing.empty) throw new Error('User already exists');
      tx.create(userRef, { auth0Id, email, name, avatar: fileUrl, quizzes: [] });});};}
```

У лістингу 3.8 представлено контролер, що обробляє запити, пов’язані зі створенням вікторини, її отриманням за ідентифікатором та генерацією додаткової відповіді за допомогою OpenAI. Кожен з маршрутів делегує виконання відповідному методу сервісу, забезпечуючи взаємодію між клієнтом і логікою обробки даних.

Лістинг 3.8 – Програмний код, що реалізує контролер для створення, перегляду вікторини та генерації відповіді за допомогою OpenAI

```
@Controller('/quizzes')
export class QuizController {
  constructor(private quizService: QuizService) {}
  @UseGuards(AuthGuard('jwt'))
  @Post('/')
  async createQuiz(@CurrentUser() { id }, @Body() dto:
CreateQuizInput) {
const quizId = await this.quizService.createQuiz(id, dto);
    return { id: quizId };}
  @Post('/generate-answer')
  async generateQuizAnswer(@Body() { question, answers }:
GenerateAnswerInput) {
    const answer = await
this.quizService.generateQuizAnswer(question, answers);
    return { answer };}
  @Get('/:quizId')
  getQuiz(@Param('quizId') quizId: string) {
    return this.quizService.getQuiz(quizId);}}
```

У лістингу 3.9 подано реалізацію сервісу, який відповідає за логіку створення нової вікторини та отримання її даних разом з пов’язаними кімнатами. Метод getQuiz здійснює запит до Firestore, перевіряє наявність вікторини за ID та додає до відповіді список кімнат. Метод createQuiz створює новий документ у колекції quizzes.

Лістинг 3.9 – Програмний код, що реалізує логіку створення, отримання вікторини та генерації відповіді за допомогою OpenAI

```
@Injectable()
export class QuizService {
  constructor(private firebaseService: FirebaseService) {}
  async getQuiz(quizId: string) {
    const quizSnap = await this.firebaseService.getFirestore().collection('quizzes').doc(quizId).get()
    if (!quizSnap.exists) throw new Error('Quiz not found')
    const roomsSnap = await this.firebaseService.getFirestore().collection('rooms').where('quizId', '==', quizSnap.id).get()
    const rooms = roomsSnap.docs.map(d => ({ id: d.id, ...d.data() }))
    return { id: quizSnap.id, ...quizSnap.data(), rooms }}
  async createQuiz(auth0Id: string, dto: CreateQuizInput): Promise<string> {
    const userSnap = await this.firebaseService.getFirestore().collection('users').where('auth0Id', '==', auth0Id).get()
    if (userSnap.empty) throw new Error('User not found')
    const userId = userSnap.docs[0].id
    const quizRef = await this.firebaseService.getFirestore().colection('quizzes').add({
      name: dto.name, description: dto.description, userId, questions: dto.questions as Question[]}) return quizRef.id}}
```

У лістингу 3.10 представлено контролер, який відповідає за управління кімнатами, пов’язаними з проходженням вікторин. Він обробляє створення нової кімнати, надсилання відповідей учасників та отримання детальної інформації про конкретну сесію. Усі запити делегуються відповідним методам сервісу, що забезпечує логічне розділення обов’язків.

Лістинг 3.10 – Програмний код, що реалізує контролер для створення кімнати, надсилання відповідей та отримання даних сесії вікторини

```
@Controller('/rooms')
export class RoomController {
  constructor(private roomService: RoomService) {}
  @Post('/:roomId/submit-answers')
  submitAnswers(@Body() dto: SubmitAnswersInput,
@Param('roomId') roomId: string) {
    return this.roomService.submitAnswer(roomId, dto)}
  @Post('/')
  async createRoom(@Body() dto: CreateRoomInput) {
    const roomId = await this.roomService.createRoom(dto)
    return { id: roomId }}
  @Get('/:roomId')
  async getRoom(@Param('roomId') roomId: string) {
    return this.roomService.getRoom(roomId)}}
```

У лістингу 3.11 представлено реалізацію сервісу, який забезпечує створення кімнати для проходження вікторини, отримання пов’язаної інформації про кімнату та вікторину, а також обробку відповідей користувача. Метод getRoom повертає повні дані про кімнату разом з відповідною вікториною, а метод createRoom додає новий документ у колекцію rooms із початковими параметрами, включаючи назву, ідентифікатор вікторини та порожній список учасників.

Основна логіка обробки відповідей реалізована у методі submitAnswer. Він виконує звірку кожної відповіді з правильною, залежно від типу запитання – одиночного вибору, множинного вибору або з текстовим ввідним полем. Для кожного типу застосовано власну перевірку коректності, після чого обчислюється загальна кількість набраних балів. Після перевірки відповідей результат додається до списку учасників кімнати у базі даних Firestore.

Лістинг 3.11 – Програмний код, що реалізує логіку створення кімнати, перевірки відповідей з різними типами запитань

```
@Injectable()
export class RoomService {
  constructor(private firebaseService: FirebaseService) {}
  async getRoom(roomId: string) {
    const roomSnap = await this.firebaseService.getFirestore().collection('rooms').doc(roomId).get()
    if (!roomSnap.exists) throw new Error('Room not found')
    const quizSnap = await this.firebaseService.getFirestore().collection('quizzes').doc(roomSnap.data().quizId).get()
    return { id: roomSnap.id, ...roomSnap.data(), quiz: { id: quizSnap.id, ...quizSnap.data() } }}
  async createRoom(input: CreateRoomInput): Promise<string> {
    const ref = await this.firebaseService.getFirestore().collection('rooms').add({
  name: input.name, quizId: input.quizId, participants: []})
    return ref.id}
  async submitAnswer(roomId: string, submitted: SubmitAnswersInput) {
    const db = this.firebaseService.getFirestore()
    const roomSnap = await db.collection('rooms').doc(roomId).get()
    if (!roomSnap.exists) throw new Error('Room not found')
    const quizSnap = await db.collection('quizzes').doc(roomSnap.data().quizId).get()
    if (!quizSnap.exists) throw new Error('Quiz not found')
    const quiz = { id: quizSnap.id, ...quizSnap.data() }
    let score = 0
    for (const a of submitted.answers) {
      const q = quiz.questions.find(q => q.id === a.questionId)
```

Продовження лістингу 3.11

```
          if (!q) throw new Error(`Question ${a.questionId} not
found`)
          switch (q.type) {
            case QuestionType.SINGLE:
              if ((a as SubmitSingleAnswer).answerId ===
q.answerId) score += q.score
              break
            case QuestionType.MULTIPLE:
              const sa = new Set((a as
SubmitMultipleAnswers).answerIds)
              const ca = new Set(q.answerIds)
              if (sa.size === ca.size && [...sa].every(id =>
ca.has(id))) score += q.score
              break
            case QuestionType.CUSTOM:
              const val = (a as SubmitCustomAnswer).answer
              if (q.answers.some(opt => opt.caseSensitive ? val
=== opt.value : val.toLowerCase() === opt.value.toLowerCase()))
score += q.score
              break}}
        await db.collection('rooms').doc(roomId).update({
          participants: firestore.FieldValue.arrayUnion({
name: submitted.name, score })})}}
```

У лістингу 3.12 реалізовано контролер, що дозволяє отримати дані поточного автентифікованого користувача на основі його Auth0-ідентифікатора. Метод getCurrentUser захищено за допомогою JWT guard, що гарантує доступ лише авторизованим користувачам. У разі успішного знаходження відповідного запису в базі даних, повертається об’єкт користувача. У випадку помилок або відсутності користувача генерується відповідне виключення з повідомленням про проблему.

Лістинг 3.12 – Програмний код, що повертає поточного автентифікованого користувача з бази даних

```
@Controller('users')
export class UserController {
  constructor(private userService: UserService) {}
  @UseGuards(AuthGuard('jwt'))
  @Get('/current')
  async getCurrentUser(@CurrentUser() { id }: CurrentUserType) {
    try {
      const user = await this.userService.getUserByAuth0Id(id)
      if (!user) throw new InternalServerErrorException("Couldn't find current user in database")
      return user
    } catch (e) {
      if (e instanceof Error && e.message === 'User not found') throw new InternalServerErrorException(e.message)
      console.log('Error:', e)
      throw new InternalServerErrorException('Something went wrong')}}}
```

У лістингу 3.13 наведено реалізацію методу, який здійснює пошук користувача в базі даних Firestore за його ідентифікатором Auth0. Якщо користувача не знайдено, метод повертає null, що дозволяє коректно обробити цей випадок на рівні контролера. У разі успішного знаходження документа з користувачем, система додатково виконує запит до колекції вікторин, щоб знайти всі створені цим користувачем вікторини. Отримані вікторини формуються у масив, кожен елемент якого включає ідентифікатор та відповідні дані. Після цього формується об’єкт користувача, який містить власні дані та масив його вікторин.

Лістинг 3.13 – Програмний код, що знаходить користувача і повертає його дані разом зі списком створених вікторин

```
@Injectable()
export class UserService {
  constructor(private firebaseService: FirebaseService, private remoteConfigService: RemoteConfigService) {}
  async getUserByAuth0Id(auth0Id: string): Promise<User | null> {
    const userSnap = await this.firebaseService.getFirestore().collection('users').where('auth0Id', '==', auth0Id).limit(1).get()
    if (userSnap.empty) return null
    const userDoc = userSnap.docs[0]
    const quizzesSnap = await this.firebaseService.getFirestore().collection('quizzes').where('userId', '==', userDoc.id).get()
    const quizzes = quizzesSnap.docs.map(d => ({ id: d.id, ...d.data() } as Quiz))
    return { id: userDoc.id, ...userDoc.data(), quizzes } as User}}
```

Реалізація серверної частини вебдодатку побудована на основі фреймворку Nest.js із використанням Firebase Firestore як хмарної бази даних, що забезпечує масштабованість, гнучкість і простоту доступу до даних.

Усі основні функціональні блоки – автентифікація користувача, створення та збереження вікторин, управління кімнатами та перевірка відповідей – реалізовані у вигляді окремих сервісів і контролерів, що гарантує модульність і зручність супроводу коду.

Загалом серверна частина забезпечує надійне та ефективне управління даними, підтримуючи всі ключові процеси, необхідні для повноцінної роботи клієнтської частини.

3.4 Інтеграція зовнішніх сервісів

Інтеграція зовнішніх сервісів є важливою частиною розробки сучасних вебдодатків, оскільки дозволяє розширити функціональність системи без потреби створювати власні рішення з нуля. У лістингу 3.14 представлено програмний код, який оголошує модуль OpenAIModule з підключеним сервісом OpenAIService.

Лістинг 3.14 – Програмний код, що визначає модуль з сервісом OpenAI для повторного використання в застосунку.

```
@Module({
  imports: [CommonModule],
  providers: [OpenAIService],
  exports: [OpenAIService],})
export class OpenAIModule {}
```

У лістингу 3.15 реалізовано сервіс OpenAIService, який ініціалізує клієнт для взаємодії з OpenAI API на основі ключа доступу з конфігурації. Метод getOpenAI надає доступ до створеного екземпляра, дозволяючи іншим сервісам використовувати функціонал OpenAI.

Лістинг 3.15 – Програмний код, що створює сервіс для доступу до OpenAI API у межах застосунку

```
@Injectable()
export class OpenAIService {
  private readonly openAI: OpenAI;
  public constructor(configService: ConfigService) {
    this.openAI = new OpenAI({
      apiKey:    configService.get('OPENAI_API_KEY')    ??
process.env.OPENAI_API_KEY,});}
  public getOpenAI() {return this.openAI;}}
```

У лістингу 3.16 наведено метод, що формує запит до OpenAI з метою згенерувати додаткову відповідь до тестового запитання. Отримана відповідь використовується як один із варіантів вибору у вікторині для підвищення її варіативності.

Лістинг 3.16 – Програмний код, що надсилає запит до OpenAI для генерації відповіді до тестового запитання

```
    async generateQuizAnswer(question: string, answers: string[]): Promise<string> {
      const content = `Plz generate one more possible answer for the following question: "${question}". I already have such answers: ${answers.map(a => `"${a}"`).join(', ')}. Do not generate any other text apart from the answer. The answer is not supposed to be right, it's going to be used as one of the options for the question`
      const response = await this.openAIService.getOpenAI().chat.completions.create({
        model: 'gpt-4o', messages: [{ role: 'system', content: 'You are a helpful assistant.' }, { role: 'user', content }]})
      return response.choices[0].message.content}
```

У лістингу 3.17 наведено логіку, що дозволяє виділити серед усіх запитань ті, які мають тип AI, сформувати з них відповідний запит до OpenAI API, а також отримати і обробити відповідь з оцінкою правильності. Для цього з масиву питань вікторини фільтруються лише ті, що передбачають оцінювання відповіді за допомогою штучного інтелекту. До кожного з них додається відповідь користувача, після чого формується prompt, який надсилається у вигляді запиту до мовної моделі GPT. У відповіді OpenAI повертає масив об'єктів, що містять ідентифікатор запитання та значення правильності відповіді, яке потім розбирається та використовується для нарахування балів учаснику вікторини.

Лістинг 3.17 – Програмний код, що фільтрує AI-запитання, надсилає їх до OpenAI для перевірки правильності відповідей та зберігає результат

```
    const aiQuestions = quiz.questions.filter(q => q.type === QuestionType.AI).map(q => ({
      ...q, answer: submittedAnswers.answers.find(a => a.questionId === q.id)}))
const aiAnswers: { questionId: string; correct: boolean }[] = []
    if (aiQuestions.length) {
      const prompt = `I'm sending you a set of questions, answers and for each question a set of additional instructions, check if an answer is correct or not. Return the data in the following way(json array): { questionId: string, correct: true }[]. Here's the set of questions and answers: ${JSON.stringify(aiQuestions)}`
      const response = await this.openAIService.getOpenAI().chat.completions.create({
    model: 'gpt-4o', messages: [{ role: 'system', content: 'You are a helpful assistant.' },{ role: 'user', content: prompt }]})
aiAnswers.push(JSON.parse(response.choices[0].message.content)
```

Загалом реалізація системи охоплює повний цикл взаємодії з користувачем – від автентифікації та створення вікторин до участі в інтерактивних кімнатах та оцінювання результатів. Вона побудована з урахуванням сучасних архітектурних підходів: клієнтська частина забезпечує гнучкий і адаптивний інтерфейс, а серверна логіка гарантує надійне управління даними й інтеграцію з зовнішніми сервісами, такими як OpenAI і Firebase. Використання штучного інтелекту дозволяє не лише динамічно формувати контент, а й перевіряти відповіді на відкриті запитання, що значно підвищує функціональні можливості системи. Така комплексна реалізація робить вебдодаток не лише ефективним засобом для проведення вікторин, а й демонструє приклад практичного застосування генеративного ШІ у навчально-розважальних середовищах.

3.5 Інтерфейс користувача

У межах цього підрозділу буде розглянуто реалізовані інтерфейси користувача, які забезпечують зручну взаємодію з вебдодатком. Основна увага приділяється ергономічності, логічній структурованості та послідовності інтерфейсних рішень, що дозволяє як організаторам, так і учасникам легко орієнтуватися у функціональності системи. Всі елементи інтерфейсу розроблено з урахуванням адаптивності, сучасних UI-практик і використання анімацій для покращення користувацького досвіду.

Після завантаження вебдодатку користувач потрапляє на головну сторінку з анімованим заголовком та кнопкою для входу в систему. Даний інтерфейс слугує початковою точкою взаємодії та реалізує логіку аутентифікації через зовнішній сервіс авторизації.

Вхід ініціюється кнопкою, яка автоматично перенаправляє користувача до сторінки входу, що забезпечує захищене з'єднання з використанням протоколу. Інтерфейс представлений у лаконічному дизайні з акцентом на основну дію на цьому етапі – авторизацію, що добре видно на головному екрані (рисунок 3.1).

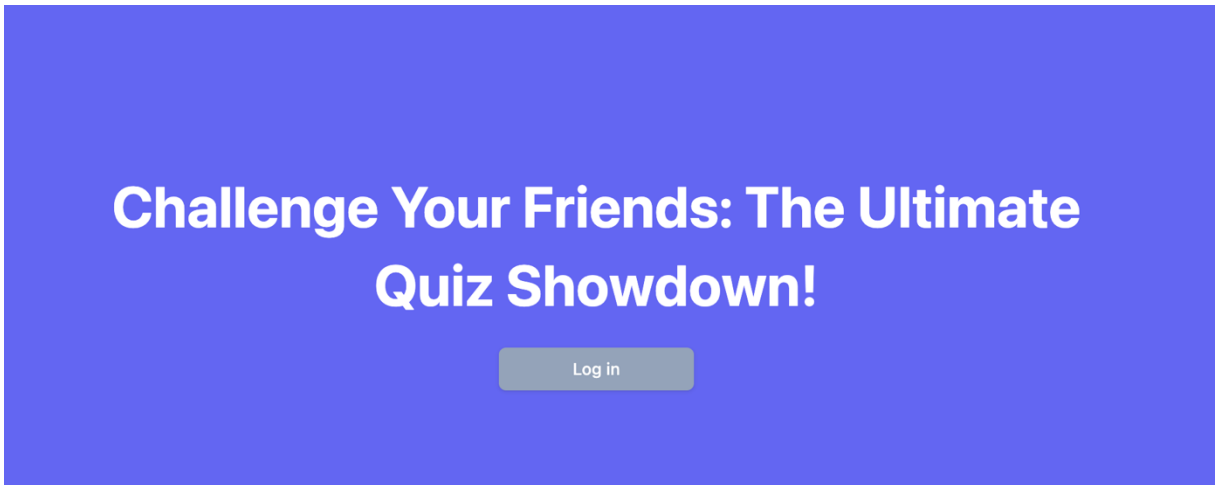

Рисунок 3.1 – Сторінка з кнопкою входу

Головна сторінка вебдодатку після авторизації містить інтерактивний дашборд, що відображає усі створені вікторини користувача. Кожна картка містить назву тесту, кількість запитань, короткий опис і кнопку для переходу до детального перегляду та керування вікториною. У правому верхньому куті розміщено кнопку для створення нової вікторини, що відкриває відповідне модальне вікно. Даний інтерфейс надає користувачеві швидкий доступ до вже створеного контенту та дозволяє ефективно управляти навчальними або розважальними матеріалами (рисунок 3.2).

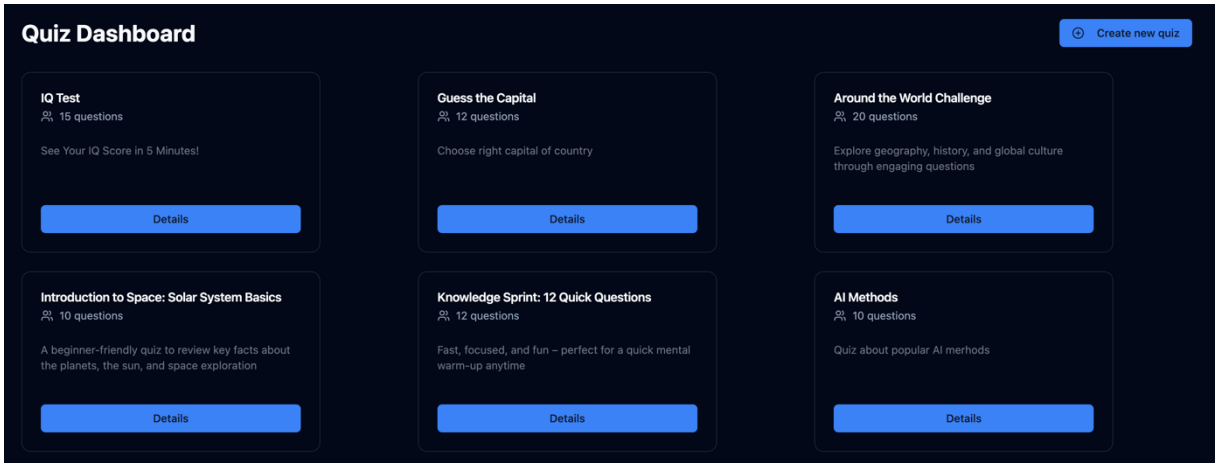

Рисунок 3.2 – Головна сторінка з вікторинами

Візуальне оформлення дашборду забезпечує зручну навігацію завдяки чіткій структурі і контрастним елементам управління. Кожен елемент має однаковий стиль подачі, що дозволяє користувачеві легко орієнтуватися серед великої кількості тестів. Завдяки короткому опису під назвою, навіть при побіжному перегляді можна зрозуміти тематику вікторини та визначити її призначення. Такий підхід сприяє швидкому прийняттю рішення користувачем щодо участі або редагування відповідного тесту.

Функціональність створення нової вікторини реалізована через інтуїтивно зрозуміле випадаюче меню, яке з’являється після натискання кнопки створення (рисунок 3.3). Користувач має можливість обрати між ручним режимом створення тесту, додаючи запитання по черзі, або

автоматизованим варіантом з використанням генеративної моделі штучного інтелекту, що суттєво пришвидшує процес.

Такий підхід дозволяє охопити різні потреби цільової аудиторії – від вчителів або викладачів, які бажають повністю контролювати структуру тесту, до тих, хто шукає швидке рішення для створення інтерактивного контенту. Візуальне розділення режимів за допомогою іконок також покращує доступність і знижує ймовірність помилкового вибору.

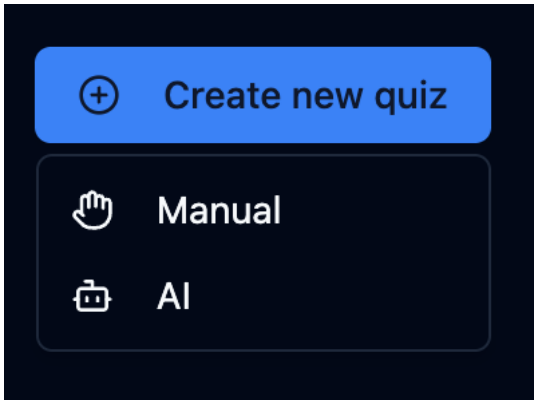

Рисунок 3.3 – Кнопка створення нової вікторини

Процес створення нової вікторини в системі може здійснюватися двома способами – вручну або автоматично за допомогою штучного інтелекту. Такий підхід забезпечує гнучкість і комфорт для різних категорій користувачів: як для тих, хто хоче мати повний контроль над кожним запитанням, так і для тих, хто бажає згенерувати базу запитань швидко за допомогою великої мовної моделі. Використання ШІ особливо актуальне у випадках, коли потрібно отримати різноманітні запитання за новою темою або протестувати знання з певної галузі без витрат часу на самостійне складання тесту.

Деталі реалізації автоматизованого способу створення вікторини ілюструє форма (рисунок 3.4). Користувач має можливість вказати тему, наприклад «History», а також додати інструкції, які допоможуть моделі сформувати запитання згідно з очікуваннями (наприклад, включити як

сучасні, так і стародавні назви міст). Інтерфейс є лаконічним і логічно структурованим, що дозволяє навіть недосвідченому користувачеві інтуїтивно виконати потрібну дію. Після натискання кнопки Generate система надсилає запит до OpenAI API та формує вікторину, яку надалі можна редагувати або відразу використати у сеансі тестування.

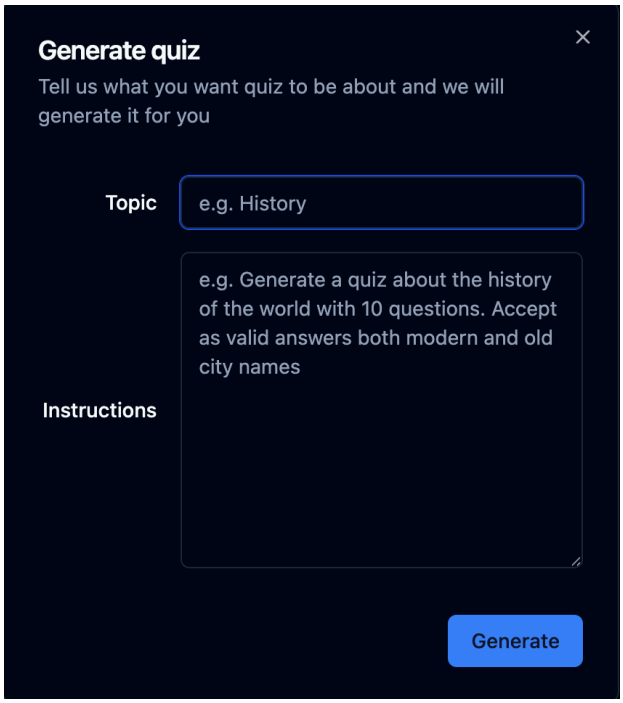

Рисунок 3.4 – Форма генерації вікторини за темою

У разі обрання ручного способу створення вікторини користувач отримує можливість самостійно визначати тип кожного запитання. Такий підхід дозволяє більш гнучко адаптувати тест до конкретних цілей, наприклад, для перевірки фактологічних знань, критичного мислення або вміння формулювати власні відповіді. Крім того, ручне створення забезпечує повний контроль над формулюванням запитань, варіантами відповідей, балами за правильні відповіді, а також критеріями оцінювання.

Вибір типу запитання реалізовано у вигляді окремого екрана з чотирма варіантами (рисунок 3.5). Користувач може додати запитання типу single choice або multiple choice, якщо хоче, щоб респондент обирав один або

кілька правильних варіантів із заданого списку. У випадку custom answer необхідно ввести правильні варіанти відповіді вручну, а, наприклад, для типу ai-powered – надати інструкції, за якими система оцінюватиме відповідь за допомогою моделі штучного інтелекту. Такий інтерфейс дозволяє будувати як прості, так і складні логічні тести, комбінуючи різні типи завдань.

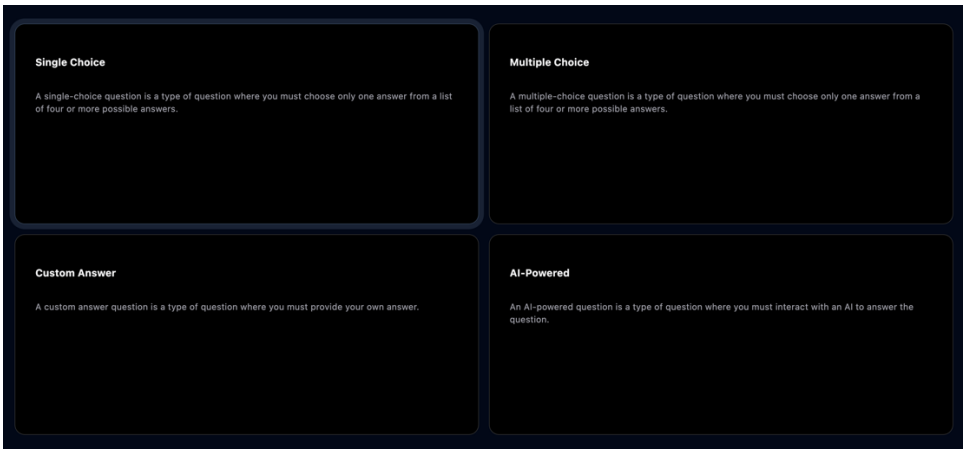

Рисунок 3.5 – Вибір типу запитання

У межах ручного створення вікторини користувач має можливість вказати запитання та додати до нього список відповідей. При цьому для кожного запитання можна визначити правильний варіант, позначивши його серед усіх запропонованих. Це забезпечує зручність у підготовці класичних тестових завдань, де лише одна відповідь є вірною, як це поширено у шкільних або вступних тестах.

На зображенні представлено приклад створення такого типу запитання (рисунок 3.6). У полі введення зазначено запитання «Хто є автором теорії відносності?», а нижче – чотири варіанти відповіді. Вірним позначено другий варіант – «Альберт Ейнштейн». Інтерфейс дає змогу легко редагувати або додавати додаткові варіанти, а також швидко переходити до наступного запитання натисканням кнопки finish.

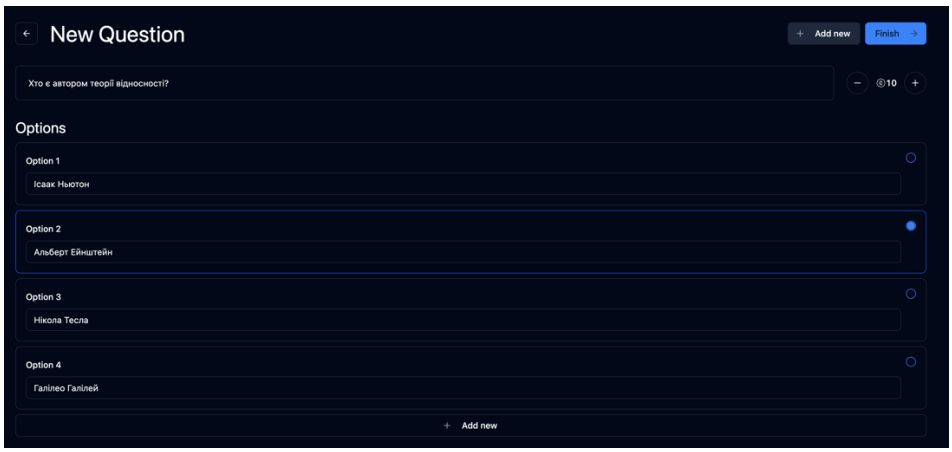

Рисунок 3.6 – Приклад створення запитання з одним правильним варіантом

У вебдодатку передбачена підтримка створення запитань із кількома правильними варіантами відповідей, що дозволяє реалізувати більш гнучкі та комплексні тести. Такий тип запитань є особливо корисним у випадках, коли для повної правильності відповіді необхідно обрати кілька елементів зі списку. Інтерфейс (рисунок 3.7) дозволяє відмітити кілька варіантів як вірні, що зручно для перевірки знань у сферах з багатоваріантною логікою.

Рисунок 3.7 – Приклад створення запитання з кількома правильними варіантами

На прикладі можна побачити запитання про країни – члени Європейського Союзу. Серед запропонованих варіантів відповіді

користувач має змогу обрати кілька правильних, наприклад, Францію та Італію. Кожен варіант супроводжується чекбоксом, за допомогою якого визначається його правильність. Завдяки цьому користувач отримує інструмент для створення якісних тестів, орієнтованих на розуміння теми, а не лише на вгадування однієї відповіді.

Для підвищення зручності створення вікторин у вебдодатку реалізовано функціональність автоматичної генерації варіантів відповідей за допомогою спеціальної кнопки. Це дозволяє користувачу зосередитися на формулюванні запитань, довіривши генерацію правдоподібних, але неправдивих варіантів системі. Далі можна побачити приклад такого запитання, у якому варіанти для відповіді генеруються автоматично за допомогою відповідних кнопок поруч з кожним полем (рисунок 3.8). Такий підхід значно пришвидшує процес створення тестів і водночас дозволяє зберегти їх якість.

Рисунок 3.8 – Приклад створення запитання з автоматично згенерованими відповідями

Щоб забезпечити більшу гнучкість при формуванні завдань, користувачеві доступна можливість створення кастомних запитань з відкритою відповіддю. Це дозволяє формулювати більш складні або творчі завдання, де відповідь не обмежується набором варіантів, а вводиться

учасником у довільній формі. Далі можна побачити приклад такого запитання, яке передбачає вільне формулювання відповіді та додаткову перевірку на відповідність (рисунок 3.9).

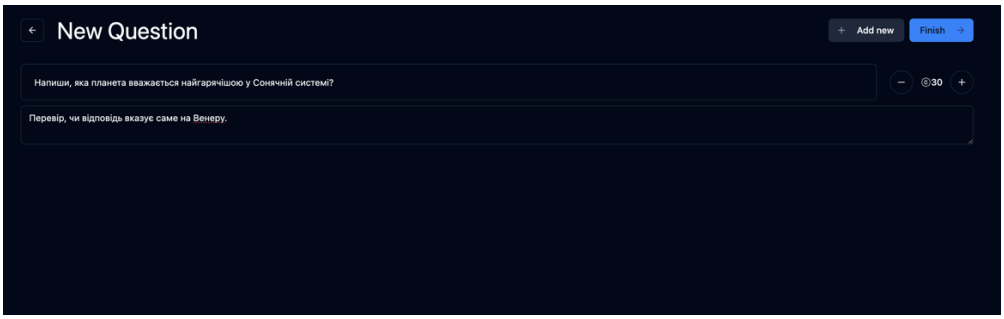

Рисунок 3.9 – Приклад створення кастомного запитання з відкритою відповіддю

Інтерфейс сторінки (рисунок 3.10) окремої вікторини містить інформацію про кількість запитань, короткий опис, а також набір функціональних кнопок для керування вікториною. Зокрема, користувач має змогу створити нову кімнату для проходження вікторини, переглянути її деталі або видалити існуючу сесію. Це забезпечує зручне адміністрування й гнучкість взаємодії з контентом.

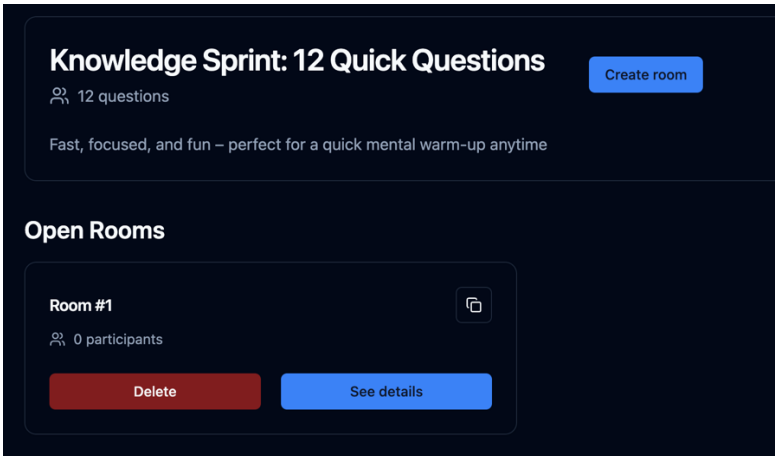

Рисунок 3.10 – Сторінка певної вікторини з кнопками управління

Після отримання посилання на створену вікторину, користувач має перед початком її проходженням ввести своє ім’я, що дозволяє ідентифікувати учасника у загальному списку результатів. Така проста та інтуїтивно зрозуміла форма (рисунок 3.11) забезпечує персоналізований досвід проходження тесту та полегшує збирання статистики.

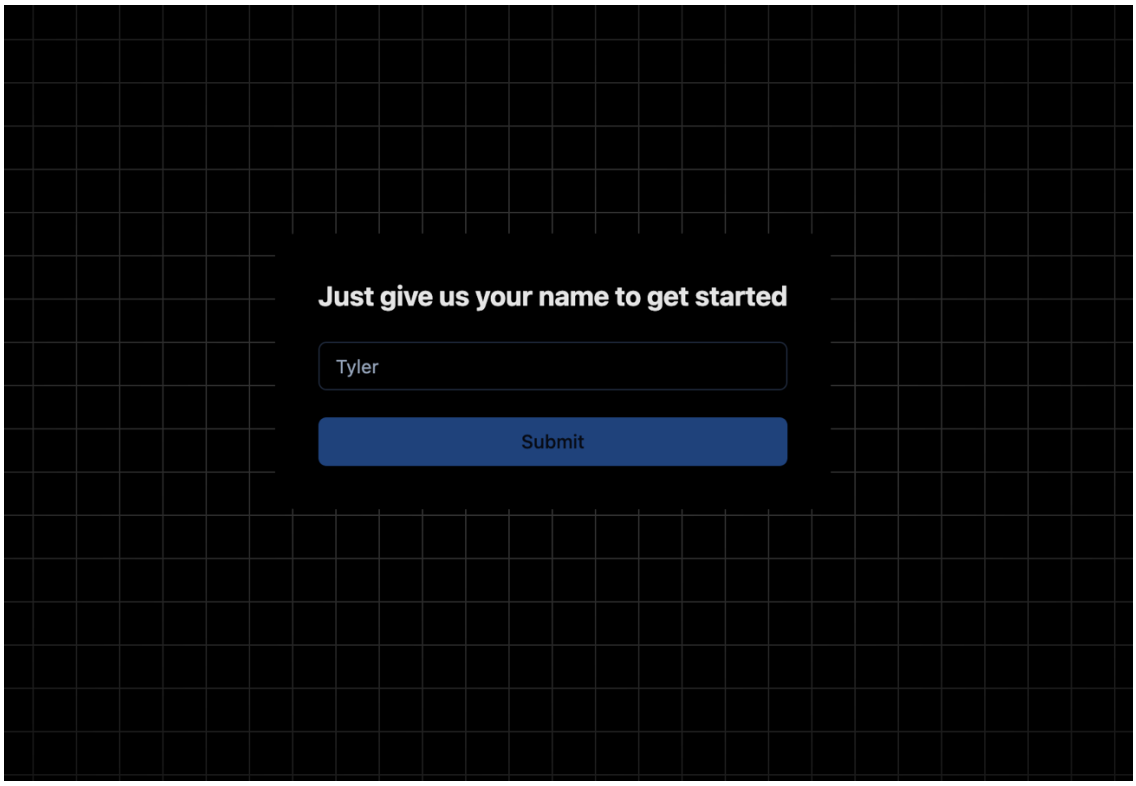

Рисунок 3.11 – Форма введення імені перед проходженням вікторини

Після введення імені користувач переходить безпосередньо до проходження вікторини, де інтерфейс запитань є мінімалістичним і зосередженим на контенті. Кожне запитання має чітко виражене формулювання, перелік варіантів відповіді у вигляді радіокнопок і кнопку переходу до наступного запитання.

Такий підхід спрощує взаємодію та дозволяє сфокусуватись на змісті запитання без зайвого відволікання. Далі можна побачити приклад такого екрану з одиночним вибором правильної відповіді (рисунок 3.12).

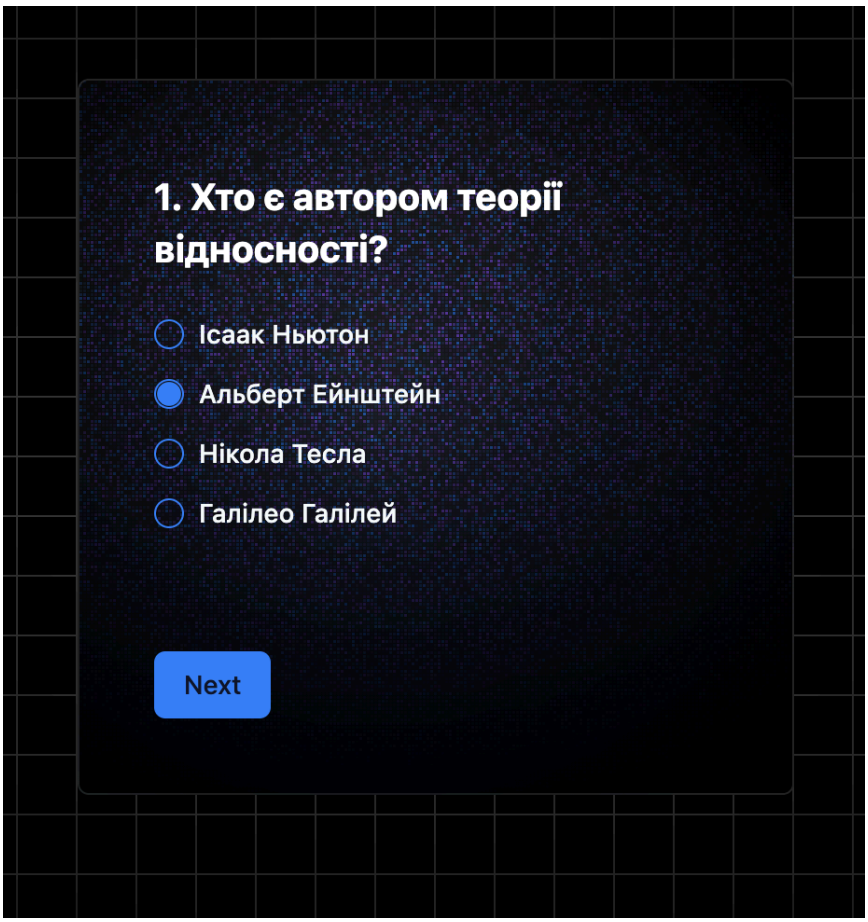

Рисунок 3.12 – Інтерфейс проходження вікторини

Після завершення проходження вікторини користувачу відображається підсумковий екран з таблицею результатів, у якій вказані імена всіх учасників та відповідна кількість набраних ними балів. Така реалізація інтерфейсу дозволяє миттєво проаналізувати ефективність відповіді, оцінити себе на фоні інших гравців та заохочує до повторного проходження тесту з метою покращення результату. Важливо, що дані учасників фіксуються одразу після завершення вікторини, що виключає можливість втрати інформації або некоректного відображення статистики. Завдяки таблиці можна також візуалізувати навчальний прогрес групи, що буде корисно у форматі навчальних сесій або під час командних змагань.

Користувацький інтерфейс на даному етапі побудований так, щоб забезпечити максимальну інформативність при збереженні візуальної простоти – чорне тло, чіткі лінії, контрастні написи та індикатори балів

створюють враження завершеного ігрового циклу. Далі можна побачити приклад такого екрану з трьома учасниками, де у відповідній таблиці виведено їхні імена та оцінки (рисунок 3.13). Такий підхід сприяє гейміфікації процесу навчання та створює позитивний змагальний ефект, що особливо важливо при використанні системи в освітньому середовищі або під час тренінгів.

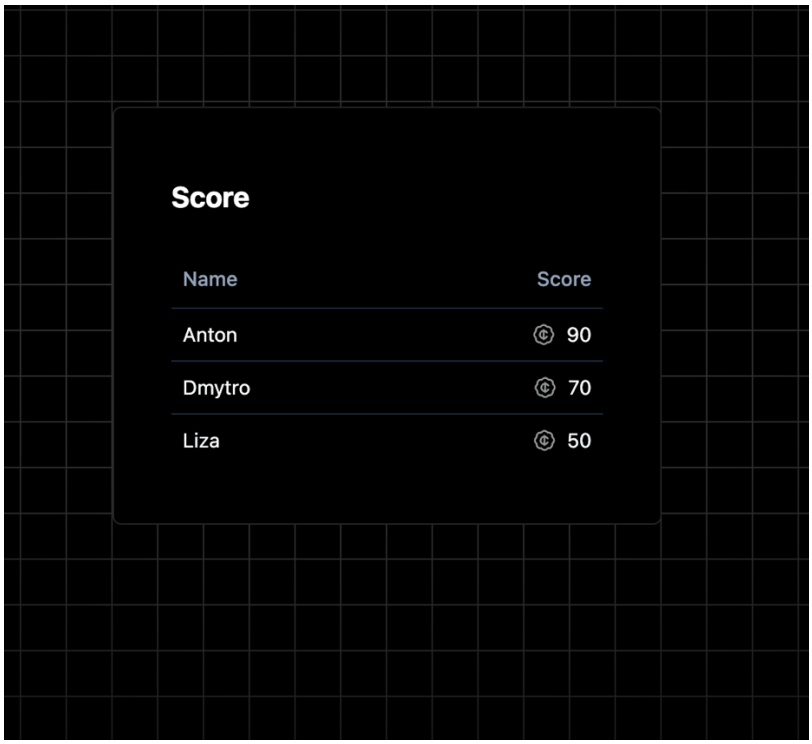

Рисунок 3.13 – Відображення підсумкових результатів учасників

Інтерфейс розробленого вебдодатку є простим, інтуїтивно зрозумілим та привабливим для користувача. Усі ключові функції – створення вікторин, генерація запитань, участь у сесіях та перегляд результатів – реалізовано з використанням зрозумілих візуальних компонентів і зрозумілих форм. Завдяки сучасному дизайну, анімаціям та зручному розташуванню елементів навігації користувачі можуть швидко орієнтуватися в системі навіть без попереднього досвіду.

3.6 Розгортання системи в хмарному середовищі

Процес розгортання створеного вебдодатку у хмарному середовищі є важливим етапом, що забезпечує доступність, масштабованість та зручне управління застосунком. У межах реалізації було обрано платформу Google Cloud Platform, яка надає великий набір інструментів для хостингу, зберігання, автентифікації, а також інтеграції з зовнішніми API. Враховуючи архітектуру застосунку, яка включає клієнтську частину, серверну логіку, базу даних Firestore та збереження медіафайлів, використання GCP дозволило централізовано керувати всіма сервісами, забезпечуючи надійну інфраструктуру для функціонування системи.

Серверна частина, реалізована на платформі Node.js з використанням фреймворку Nest.js, була обгорнута в Docker-контейнер для забезпечення ізоляції середовища та простоти в розгортанні. Docker-образ було завантажено до Google Container Registry, після чого він був розгорнутий за допомогою Google Cloud Run. Цей сервіс дозволяє автоматично масштабувати контейнер залежно від навантаження та керувати версіями застосунку без необхідності ручного налаштування серверів. Cloud Run підтримує інтеграцію з іншими сервісами GCP, такими як Firestore, Cloud Storage та Secret Manager, що дозволило централізовано зберігати облікові дані до OpenAI API та інших конфіденційних змінних середовища.

Клієнтська частина застосунку була побудована на основі фреймворку React з використанням сучасного інструменту Vite для білду. Після збирання статичних файлів застосунок було розгорнуто на Google Firebase Hosting. Цей сервіс забезпечує швидке та надійне обслуговування вебресурсів з підтримкою, що гарантує високу продуктивність при доступі до додатку з різних регіонів. Firebase Hosting також дозволив просто реалізувати маршрутизацію для SPA, а також автоматично обробляє HTTPS-сертифікати, що забезпечує безпечне з'єднання з користувачем.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було повністю реалізовано завдання з розробки вебдодатку для створення, проходження та оцінювання вікторин з інтеграцією штучного інтелекту. Було побудовано повнофункціональний вебдодаток з чітко вираженим розподілом клієнтської і серверної частин, зручним та інтуїтивним інтерфейсом для користувачів різних ролей – організатора та учасника, а також реалізовано систему генерації варіантів відповідей і перевірки складних типів запитань за допомогою моделі GPT від OpenAI. Інфраструктура побудована на базі Google Cloud Platform, що забезпечує високу надійність, масштабованість і безпеку. Усі ключові функціональні блоки системи протестовані та підтвердили свою коректну роботу в реальному режимі. Кількісно, реалізовано підтримку чотирьох типів запитань, збереження даних у хмарній базі Firestore, повну обробку сесій вікторин, результати яких зберігаються у режимі реального часу. Якісно ж, було досягнуто гнучкої та зручної взаємодії між користувачем та системою, що підтверджується ергономічністю інтерфейсу та позитивним зворотним зв’язком від тестувальників.

Порівняння реалізованого рішення з існуючими аналогами демонструє низку переваг. Серед найпопулярніших сервісів можна виокремити такі як Kahoot, Quizizz, Google Forms. Усі вони мають обмежену або зовсім відсутню підтримку інтеграції зі штучним інтелектом. Розроблений вебдодаток дозволяє не лише створювати запитання вручну, а й генерація повноцінної вікторини за заданою темою, також автоматично генерувати додаткові варіанти відповідей або перевіряти відкриті відповіді на основі гнучких інструкцій до ШІ, що значно розширює його функціональність порівняно з аналогами. Крім того, система не потребує встановлення та може бути використана з будь-якого пристрою, що підтримує сучасний браузер, що робить її універсальною. Відкрита

архітектура дозволяє просто додавати нові типи запитань, оновлювати інтерфейс, а також масштабувати проект у майбутньому під додаткові ролі користувачів або контексти використання, наприклад у навчальних закладах, на тренінгах чи в корпоративному середовищі.

Перспективи подальшого розвитку застосунку є досить широкими. Насамперед, доцільно розширити інтеграцію з OpenAI API для повноцінної генерації всіх запитань вікторини на основі навчального матеріалу. Крім того, можливо впровадити адаптивну систему складності, яка на основі відповідей учасника автоматично пропонує більш складні або прості запитання, що значно підвищить ефективність навчання. Також важливим напрямом є впровадження аналітичного модуля для організаторів, який дозволить оцінювати статистику по відповідях, визначати типові помилки та формувати рекомендації для подальших занять. Важливим етапом також є створення мобільного застосунку або адаптація поточного інтерфейсу під мобільні пристрої для забезпечення ще більш широкого доступу до системи. Нарешті, можна реалізувати можливість колективного створення тестів декількома організаторами або проведення вікторин у форматі реального часу з рейтинговими таблицями, що сприятиме гейміфікації процесу і залученню більшої кількості користувачів.

Таким чином, результати кваліфікаційної роботи не лише підтвердили актуальність поставленої задачі, а й дозволили створити технічно завершене та готове до використання програмне рішення з чіткими перевагами над існуючими аналогами. Реалізований вебдодаток відкриває широкі можливості для подальших досліджень та практичного використання у сфері цифрової освіти, самооцінки знань та інтерактивного навчання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Almarashdeh I. A. Sharing instructors experience of learning management system: A technology perspective of user satisfaction in distance learning course. *Computers in Human Behavior*. 2016. Vol. 63. P. 249–255. URL: https://doi.org/10.1016/j.chb.2016.05.014.

2. Balakrishnan V., Gan C. L. Students' learning styles and their effects on the use of social media technology for learning. *Telematics and Informatics*. 2016. Vol. 33, no. 3. P. 808–821. URL: https://doi.org/10.1016/j.tele.2015.12.004.

3. Basuki Y., Hidayati Y. Kahoot! or Quizizz: the students' perspectives. *Proceedings of the 3rd English Language and Literature International Conference (ELLiC)*. 2019. P. 89–94. URL: http://eprints.ums.ac.id/71408/(date of access: 20.03.2025).

4. Bicen H., Kocakoyun S. Perceptions of students for gamification approach: Kahoot as a case study. *Interactive Learning Environments*. 2018. Vol. 26, no. 8. P. 1091–1104. URL: https://doi.org/10.1080/10494820.2017.1337039.

5. Fajar M. The effect of using quizizz application on students' motivation in learning English. *Journal of English Language Teaching and Learning*. 2022. Vol. 3, no. 1. P. 24–30. URL: https://ejournal.unib.ac.id/index.php/elt/article/view/19845(date of access: 04.04.2025).

6. OpenAI. OpenAI API documentation. URL: https://platform.openai.com/docs/(date of access: 10.04.2025).

7. Shyshkina M. P., Kohut U. P., Popel M. V. Implementation of augmented reality tools in the training of future computer science teachers: Theoretical aspects. *CEUR Workshop Proceedings*. 2020. P. 548–559. URL: http://ceur-ws.org/Vol-2732/20200548.pdf(date of access: 20.04.2025).

8. Slutskin A. Online quiz platforms for educational purposes: A comparative study of usability and effectiveness. *International Journal of*

Educational Technology in Higher Education. 2021. Vol. 18, no. 1. P. 46. URL: https://doi.org/10.1186/s41239-021-00273-1.

9. Students' perception of Kahoot!'s influence on teaching and learning / S. A. Licorish et al. *Research and Practice in Technology Enhanced Learning*. 2018. Vol. 13, no. 1. P. 9. URL: https://doi.org/10.1186/s41039-018-0078-8.

10. Widodo A., Mustofa M. Enhancing students' learning through interactive quizzes: An analysis of Quizizz implementation. *Journal of Educational Research and Evaluation*. 2021. Vol. 10, no. 3. P. 505–514. URL: https://doi.org/10.23887/jere.v10i3.36382.

11. Zainuddin Z., Attaran M. Malaysian students' perceptions of flipped classroom: A case study. *Innovations in Education and Teaching International*. 2016. Vol. 53, no. 6. P. 660–670. URL: https://doi.org/10.1080/14703297.2015.1102079.

12. Chen C. M., Tsai Y. N. Interactive quiz game system based on digital learning analytics to enhance learning motivation and performance. *Interactive Learning Environments*. 2020. Vol. 28, no. 6. P. 764–781. URL: https://doi.org/10.1080/10494820.2018.1548483.

13. Cloud G. Cloud Run Documentation. 2024. URL: https://cloud.google.com/run/docs(date of access: 04.05.2025).

14. De Boer J., Knezek G. Student perceptions of automated quiz generation using natural language processing. *International Journal of Artificial Intelligence in Education*. 2019. Vol. 29, no. 3. P. 347–370. URL: https://doi.org/10.1007/s40593-019-00178-w.

15. Firebase. Cloud Firestore Documentation. 2024. URL: https://firebase.google.com/docs/firestore(date of access: 02.05.2025).

16. Iftene A., Trăuşan-Matu Ş. Gamification and web-based quizzes for technology-enhanced formative assessment. *International Journal of Emerging Technologies in Learning (iJET)*. 2020. Vol. 15, no. 2. P. 112–127. URL: https://doi.org/10.3991/ijet.v15i02.11523.

17. Neumann D. L., Neumann D. L. The use of Kahoot in higher education: A review of the literature. *Teaching with Technology*. 2019. Vol. 6, no. 3. P. 1–13. URL: https://doi.org/10.37074/jalt.2019.6.3.8.

18. Shapiro A. M., Steiner C. A. Using AI-driven question generation to improve learning outcomes in online assessments. *Journal of Educational Computing Research*. 2021. Vol. 59, no. 5. P. 953–975. URL: https://doi.org/10.1177/07356331211018734.

19. Sulaiman S., Aziz M. S. A. The effectiveness of Quizizz as an assessment tool in online classes. *Journal of Technology and Online Education*. 2020. Vol. 8, no. 1. P. 13–20. URL: https://doi.org/10.5281/zenodo.3768814.

20. Wibowo F. R., Cahyaningsih S. The impact of AI-generated quizzes on students' critical thinking skills. *Education and Information Technologies*. 2023. Vol. 28. P. 3379–3394. URL: https://doi.org/10.1007/s10639-023-11678-7.

21. Zhang Y., Yu S. AI in education: Automatic quiz generation from textual materials using deep learning. *Computers \& Education: Artificial Intelligence*. 2022. Vol. 3. P. 100067. URL: https://doi.org/10.1016/j.caeai.2022.100067.