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)

Гейміфікована платформа для вивчення мов
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-3
Ольга Попівненко
(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 Попівненко Ользі Олегівні
(прізвище, ім'я, по батькові)

1. Тема роботи Гейміфікована платформа для вивчення мов

затверджена наказом університету від 19 травня 20 25 р. № 378Ст
2. Термін подання студентом роботи до екзаменаційної комісії 17 червня 20 25 р.
3. Вихідні дані до роботи Офіційні документації до React.js, Joy UI, TypeScript, Firebase, Google Cloud Storage, Nest.js, Auth0 та Redux Toolkit.

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	22.05.2025	виконано
5	Реалізація гейміфікованої платформи	23.05.2025	виконано
6	Оформлення пояснювальної записки	03.06.2025	виконано
7	Захист	17.06.2025	виконано

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

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

РЕФЕРАТ

Пояснювальна записка: 97 с., 18 рис., 1 дод., 21 джерело.

ГЕЙМІФІКАЦІЯ, ІНТЕРАКТИВНЕ НАВЧАННЯ, МОВНЕ СЕРЕДОВИЩЕ, ОСВІТНІЙ ВЕБДОДАТОК, ТЕСТУВАННЯ ЗНАНЬ, CLIENT-SERVER ARCHITECTURE, NEST.JS, REACT.

Об'єктом дослідження є процес вивчення іноземних мов за допомогою інтерактивних вебзастосунків.

Предметом дослідження виступають гейміфіковані механізми, що підвищують ефективність засвоєння лексичних і граматичних конструкцій у цифровому середовищі.

Метою роботи є розробка інтерактивної навчальної платформи з тестовими завданнями та системою нагород, адаптованої для багатомовної аудиторії.

Методами дослідження стали аналіз аналогів, проектування архітектури вебзастосунку, реалізація серверної та клієнтської частин за допомогою сучасних технологій, а також проведення тестування.

У результаті створено багатофункціональну платформу з унікальним гейміфікованим підходом, яка забезпечує адаптивне навчання, підтримку кількох мов та мотиваційний механізм, рекомендовану для використання у школах, онлайн-курсах і самостійній підготовці.

ABSTRACT

Bachelor’s thesis contains: 97 pp., 18 fig., 1 ann., 21 references.

CLIENT-SERVER ARCHITECTURE, GAMIFICATION, INTERACTIVE LEARNING, KNOWLEDGE TESTING, LANGUAGE ENVIRONMENT, NEST.JS, REACT, WEB EDUCATIONAL APPLICATION.

The object of the research is the process of learning foreign languages using interactive web applications.

The subject of the research is gamified mechanisms that enhance the effectiveness of acquiring lexical and grammatical structures in a digital environment.

The aim of the work is to develop an interactive educational platform with test-based tasks and a reward system, adapted for a multilingual audience.

The research methods include analysis of analogs, architectural design of the web application, implementation of the server and client sides using modern technologies, and usability testing.

As a result, a multifunctional platform was created with a unique gamified approach that ensures adaptive learning, multilingual support, and a motivational system, recommended for use in schools, online courses, and self-study.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Application Programming Interface – прикладний програмний інтерфейс;

CI/CD – Continuous Integration / Continuous Delivery – безперервна інтеграція та доставка;

DOM – Document Object Model – об'єктна модель документа;

DTO – Data Transfer Object – об'єкт передавання даних;

HTTP – Hypertext Transfer Protocol – протокол передачі гіпертексту;

MUI – Material UI – бібліотека інтерфейсних компонентів;

REST – Representational State Transfer – передача репрезентативного стану;

SDK – Software Development Kit – комплект розробника програмного забезпечення;

SPA – Single Page Application – односторінковий застосунок;

UI – User Interface – інтерфейс користувача;

UID – Unique Identifier – унікальний ідентифікатор;

URL – Uniform Resource Locator – уніфікований локатор ресурсу;

UX – User Experience – користувацький досвід.

ВСТУП

У сучасному цифровому середовищі спостерігається стрімке зростання інтересу до вивчення іноземних мов за допомогою онлайн-інструментів. Розвиток інформаційних технологій спричинив значні зміни у підходах до навчання, зробивши його більш доступним, інтерактивним та персоналізованим. Зокрема, індустрія мовної освіти активно адаптується до нових умов, використовуючи інноваційні методи для покращення мотивації та залучення користувачів. Традиційні курси поступово поступаються місцем платформам, які надають можливість вивчати мову в зручному темпі, незалежно від місця перебування. Однією з найпомітніших тенденцій останніх років стало впровадження гейміфікації у навчальний процес, що дозволяє поєднувати здобуття знань із елементами гри.

Проблема недостатньої мотивації у користувачів під час вивчення мовної граматики та лексики є актуальною і залишається викликом навіть для найпопулярніших платформ. Часто навчання сприймається як обов’язок або рутина, а не як захопливий процес. Багато сучасних освітніх рішень надають або надто формальний контент, або недостатню структуру для поступального засвоєння матеріалу. Водночас користувачі очікують не лише якісного контенту, але й простого, зручного інтерфейсу, системи заохочення, гнучкості у виборі тем і можливості відстеження прогресу. Це створює потребу у вдосконаленні існуючих рішень або розробці нових систем, які враховують як методичні вимоги, так і звички сучасних користувачів цифрових технологій.

Застосування гейміфікації в освіті довело свою ефективність у різних контекстах, особливо в роботі з молоддю та дітьми. Ігрові механіки, такі як нагороди, рівні, бали, рейтинги й досягнення, формують додаткову мотивацію для користувачів і сприяють регулярній взаємодії з навчальним матеріалом. Вони створюють відчуття прогресу, викликають позитивні емоції та стимулюють досягнення нових цілей. У поєднанні з добре

структурованими навчальними блоками гейміфікація дозволяє не лише залучити користувача до платформи, а й утримати його на довгостроковій основі. Це особливо важливо у випадку з мовним навчанням, яке вимагає постійної практики й повторення для закріплення знань.

Запропонована у межах кваліфікаційної роботи система передбачає створення веб-застосунку, який реалізує гейміфікований підхід до вивчення іноземних мов на основі проходження тематичних тестів. Основна мета такої системи – забезпечити ефективне, структуроване та водночас цікаве навчання з використанням ігрових механізмів. Користувачам буде доступна авторизація, персональний профіль, прогрес тестування, система нагород та можливість розблоковувати нові теми. У платформі передбачена підтримка кількох мов, що розширює потенційну аудиторію та робить застосунок універсальним засобом мовної практики.

Актуальність даної роботи полягає в необхідності створення доступного, мотиваційного та функціонально повного ресурсу для вивчення мов у форматі гри. Враховуючи значний попит на неформальне навчання, розширення можливостей самонавчання та загальну цифровізацію освіти, запропонований веб-застосунок може знайти широке застосування серед школярів, студентів, працівників, які вивчають іноземні мови самостійно, а також серед освітніх установ, які впроваджують сучасні засоби дистанційного навчання. Система може бути легко масштабована, адаптована під інші мови, доповнена новими функціями та використана як у формальному, так і в неформальному освітньому середовищі.

Таким чином, створення гейміфікованої платформи для вивчення мов є своєчасним і доцільним кроком, який дозволяє ефективно поєднати навчальні цілі з інструментами мотивації. Робота має не лише теоретичне значення для дослідження підходів до цифрового навчання, але й практичну цінність як приклад реальної реалізації повноцінного застосунку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Огляд сучасного стану цифрової мовної освіти

У сучасному світі цифрові технології дедалі активніше впроваджуються в різні сфери життя, зокрема й у процес навчання. Особливо це стосується вивчення іноземних мов, де використання мобільних додатків, веб-платформ та інтерактивних сервісів стало звичним явищем. Цифрова мовна освіта охоплює широкий спектр інструментів – від онлайн-курсів і словників до повноцінних освітніх екосистем, які пропонують персоналізований підхід до навчання.

Зміна парадигми в освітньому процесі обумовлена не лише розвитком технологій, а й зростанням потреби у гнучких і доступних методах отримання знань. Традиційні форми навчання вже не відповідають очікуванням багатьох користувачів, особливо молоді, яка шукає ефективні, інтерактивні та візуально привабливі засоби для освоєння нової інформації. У відповідь на ці запити розвиваються інноваційні цифрові рішення, які поєднують навчання з елементами розваги.

Серед найбільш динамічно зростаючих напрямів цифрової освіти можна виділити гейміфікований підхід. Його суть полягає у використанні ігрових механік для підвищення залученості та мотивації учнів. Такий підхід дозволяє не лише зробити навчальний процес цікавішим, а й забезпечити повторюваність дій, яка є ключовою для запам’ятовування та автоматизації мовних навичок.

Крім того, цифрова мовна освіта дозволяє масштабувати навчальні продукти та адаптувати їх під індивідуальні потреби користувачів. За допомогою сучасних веб-інтерфейсів і аналітики розробники можуть створювати зручні та ефективні інструменти для вивчення мови, які підтримують різні моделі навчання.

1.1.1 Цифрові технології у вивченні іноземних мов

Цифрові технології суттєво змінили підхід до вивчення іноземних мов, відкривши нові можливості для інтерактивного та самостійного навчання. Завдяки розвитку мережевих інфраструктур та мобільних пристроїв, користувачі можуть навчатися у зручний для себе час, у власному темпі та з будь-якого місця. Це створило умови для більш широкого доступу до мовної освіти.

Однією з ключових переваг цифрових технологій є можливість мультимедійної подачі матеріалу. Аудіо- та відеоконтент, інтерактивні вправи, вбудовані перекладачі та словники сприяють кращому засвоєнню інформації та розвитку навичок сприймання і мовлення. Такі елементи дозволяють наблизити навчальний процес до реального мовного середовища.

Також важливою характеристикою цифрового підходу є гнучкість структури навчального матеріалу. Користувач може обирати послідовність проходження тем, адаптувати рівень складності завдань та зосередитися на тих аспектах мови, які є для нього найважливішими. Це особливо цінно для дорослих користувачів і людей з обмеженим часом.

Окрім цього, цифрові платформи дозволяють ефективно відстежувати прогрес навчання. Завдяки вбудованій аналітиці, системи можуть зберігати результати, нагадувати про необхідність повторення матеріалу, а також формувати рекомендації на основі попередньої активності. Такий підхід покращує мотивацію та підтримує регулярність занять.

1.1.2 Актуальність інтерактивного навчання у форматі гри

Інтерактивне навчання у форматі гри набуло значної популярності у сфері цифрової освіти, особливо у вивченні мов. Такий підхід дозволяє поєднати навчальний процес із розважальними елементами, що позитивно

впливає на рівень залученості користувачів. Гра, як природна форма діяльності людини, особливо ефективна для освоєння нової інформації в ненав’язливому середовищі.

Ігровий формат стимулює допитливість, викликає емоційну реакцію і створює відчуття прогресу. Завдяки цьому користувачі частіше повертаються до навчання, прагнучи досягти нових цілей або подолати черговий рівень. Це контрастує з традиційними методами навчання, які часто сприймаються як рутинні або складні.

Серед основних елементів, що формують інтерактивне навчання, можна виділити віртуальні нагороди, бали, змагання з іншими учасниками, рівні складності та різноманітні сценарії. Такі механізми створюють ефект гри, але при цьому зберігають навчальну цінність. Вони підсилюють внутрішню мотивацію та сприяють утворенню позитивного підкріплення.

Інтерактивність також виявляється у способі подачі матеріалу. На відміну від пасивного читання або прослуховування, користувач активно взаємодіє з контентом – вибирає відповіді, складає речення, отримує миттєвий зворотний зв'язок. Це сприяє глибшому розумінню теми та кращому закріпленню знань.

У результаті інтерактивне навчання у форматі гри не лише робить освітній процес цікавішим, а й забезпечує умови для систематичного й ефективного засвоєння матеріалу. Завдяки цьому підходу користувачі можуть легше долати психологічний бар'єр у вивченні мов і розвивати впевненість у власних знаннях.

1.1.3 Застосування гейміфікації в освітніх технологіях

Гейміфікація в освітніх технологіях передбачає інтеграцію ігрових елементів у навчальний процес з метою підвищення зацікавленості та мотивації користувачів. Такий підхід активно використовується у цифрових додатках для вивчення мов, де навчання перетворюється на послідовність

завдань з винагородами, досягненнями та рейтингами. Гейміфікація дозволяє підтримувати інтерес до навчання навіть у разі повторення схожих за змістом тем.

Основні елементи гейміфікації включають бали, рівні, бейджі, таблиці лідерів та систему завдань. Вони формують механізм зовнішньої мотивації, але з часом можуть сприяти розвитку внутрішнього бажання навчатись. Завдяки такому підходу користувачі отримують відчуття прогресу та досягнення мети, що є важливим психологічним чинником у довготривалому навчанні.

Сучасні освітні платформи дедалі частіше орієнтуються на створення повноцінного гейміфікованого середовища, де кожна дія користувача має результат, зворотний зв’язок і потенціал для винагороди. Це не лише підвищує ефективність навчання, а й робить процес більш емоційно насиченим і привабливим для широкого кола користувачів різного віку.

1.2 Огляд конкурентних рішень

Сфера цифрового навчання іноземних мов сьогодні представлена великою кількістю рішень, які активно конкурують між собою за увагу користувача. Більшість із них пропонують зручний інтерфейс, доступ до навчального контенту на різних мовах і варіативність у методах подачі матеріалу. Завдяки цьому ринок мовних застосунків постійно зростає та розвивається.

Серед доступних інструментів можна виділити як універсальні платформи з широким функціоналом, так і вузькоспеціалізовані додатки, орієнтовані на конкретні аспекти вивчення мови. Кожен із таких продуктів має власну філософію побудови навчального процесу та використовує різні механізми мотивації, зокрема гейміфікацію, адаптивність або соціальну взаємодію.

Аналіз існуючих рішень дозволяє виявити як сильні сторони підходів, так і недоліки, що можуть заважати ефективному навчанню. Деякі платформи зосереджені на пасивному засвоєнні матеріалу, інші мають обмежену кількість мов або тематичних модулів. Вивчення цих характеристик є важливим етапом при проектуванні нових систем.

В умовах високої конкуренції важливо не лише повторювати наявні рішення, а й шукати нові підходи, які враховують потреби сучасних користувачів. Це створює підґрунтя для інноваційних продуктів, які здатні поєднувати ефективні освітні методики з привабливим і зручним користувацьким досвідом.

1.2.1 Найпопулярніші гейміфіковані додатки

Одним із найвідоміших додатків для вивчення мов є Duolingo. Він пропонує навчання у формі коротких інтерактивних вправ, побудованих за принципом гри. Користувачі виконують завдання, заробляють бали та проходять рівні, що створює відчуття прогресу. Duolingo також активно використовує нагадування і щоденні цілі, щоб підтримувати регулярність навчання.

Ще одним популярним рішенням є Memrise. Цей застосунок зосереджений на запам’ятовуванні слів і фраз за допомогою мнемотехнік та повторення з інтервалами. Відмінною рисою платформи є відеофрагменти з носіями мови, що дозволяє краще зрозуміти контекст використання фраз і покращити сприймання на слух.

Babbel орієнтується на більш структуроване навчання з граматичними поясненнями, вправами та контекстом використання. Його курси створюються професійними лінгвістами та мають чітку логіку побудови. Babbel підходить тим, хто віддає перевагу класичному підходу до навчання з детальними інструкціями.

Mondly пропонує вивчення понад 30 мов і використовує як текстові, так і голосові вправи. Особливістю додатку є віртуальний чат-бот, який дозволяє практикувати діалоги. Крім того, платформа має окремі модулі для дітей та бізнес-користувачів, що розширює її функціональність.

Busuu поєднує в собі функції онлайн-курсу та соціальної мережі. Користувачі можуть проходити уроки, а також отримувати зворотний зв'язок від носіїв мови. Такий підхід створює додаткову мотивацію і сприяє розвитку навичок письма та мовлення у реальному спілкуванні.

LingQ є платформою, яка орієнтується на читання і слухання автентичних матеріалів. Вона дозволяє створювати персоналізовані курси на основі інтересів користувача, використовуючи статті, подкасти і відео. Цей підхід підходить тим, хто вже має базовий рівень і прагне розвивати мову в реальному контексті.

1.2.2 Аналіз сильних та слабких сторін конкурентів

Серед сильних сторін Duolingo можна виділити зручний інтерфейс, привабливий дизайн і добре реалізовану систему мотивації. Щоденні цілі, віртуальні нагороди, таблиці лідерів і рівні залучають користувачів і сприяють регулярному навчанню. Однак при цьому платформа не завжди забезпечує глибоке розуміння граматичних структур і часто обмежується зазубрюванням фраз.

Memrise вирізняється якісним контентом з використанням відео з носіями мови, що робить навчання ближчим до реального спілкування. Мнемотехнічні методи та повторення з інтервалами є ефективними для запам'ятовування нової лексики. Водночас користувачі відзначають обмеженість граматичних пояснень і брак інтерактивних сценаріїв.

Babbel має сильну сторону у вигляді граматично обґрунтованих курсів, які відповідають загальноприйнятим рівням володіння мовою. Курси розроблені лінгвістами й орієнтовані на поступове ускладнення

матеріалу. Проте платформа менш приваблива для молодіжної аудиторії через формальний стиль і відсутність ігрових елементів.

Mondly приваблює широким вибором мов і голосовими функціями, що дозволяють тренувати вимову. Віртуальний чат-бот додає елемент діалогу, який підвищує практичну цінність додатку. Але іноді користувачі стикаються з поверхневістю тем і відсутністю глибокого пояснення правил.

Busuu має перевагу у вигляді спільноти користувачів, які можуть допомагати одне одному. Завдяки цьому створюється ефект живого середовища для мовної практики. Основним недоліком є часткова закритість функцій для безкоштовних користувачів, що обмежує доступ до повного навчального досвіду.

LingQ дозволяє створювати персоналізовані навчальні матеріали, що особливо корисно для просунутих учнів. Величезна база автентичного контенту дає змогу зануритися в мову природним шляхом. Однак платформа може бути складною для новачків через відсутність структурованих уроків і складність навігації.

1.2.3 Нішеві можливості для інновації

Аналіз існуючих платформ для вивчення мов показує, що більшість з них зосереджуються або на гейміфікації, або на глибокому мовному аналізі, рідко поєднуючи обидва підходи. Це відкриває можливість для створення продукту, який би ефективно балансував між ігровим залученням та якісним навчальним контентом. Користувачі шукають інструменти, які одночасно є зручними, приємними у використанні та результативними з погляду мовного прогресу.

Однією з нішевих можливостей є створення платформи з розширеною системою досягнень, що мотивує не тільки на кількість, але й на різноманітність вивчених тем. Такий підхід дозволяє підтримувати інтерес користувача у довготривалій перспективі та заохочує до комплексного

опанування мови. Окрема увага може приділятися темам, які часто ігноруються у стандартних курсах, наприклад, словнику емоцій чи фразам для подорожей.

Ще одним потенціалом для інновації є акцент на мульти-мовності платформи з можливістю гнучкого перемикання між мовами, що дозволяє користувачу навчатися одночасно кільком мовам у зручному для себе порядку. Це стане у пригоді не лише поліглотам, а й тим, хто живе в багатомовному середовищі або часто змінює локацію.

Також перспективною є ідея більш глибокої персоналізації навчального процесу без складних алгоритмів штучного інтелекту. Просте збереження прогресу, вибір улюблених тем і адаптація навчальних маршрутів під інтереси користувача можуть значно покращити якість взаємодії з додатком. Такі особливості роблять платформу ближчою до потреб конкретної людини.

1.3 Аналіз потреб користувачів

Успішність цифрового освітнього продукту значною мірою залежить від глибокого розуміння потреб користувачів. Різні категорії користувачів мають різні очікування від платформи для вивчення мов – для когось важлива простота інтерфейсу, для інших мотиваційна система, а хтось шукає швидкий результат у конкретній сфері, наприклад, подорожах або роботі. Саме тому аналіз потреб є ключовим етапом проектування системи.

Загальні тенденції показують, що користувачі все частіше надають перевагу гнучким, адаптивним та візуально привабливим рішенням. Вони очікують швидкого старту без складної реєстрації, зрозумілих інструкцій, інтерактивних вправ та можливості самостійно обирати темп навчання. Важливу роль відіграє також елемент гри, який дозволяє зберігати мотивацію упродовж усього процесу.

Крім функціональних потреб, користувачі мають і специфічні очікування щодо контенту. Багатьом важливо, щоб приклади були актуальними та близькими до реального життя, а також щоб теми відповідали їх інтересам і ситуаціям, з якими вони стикаються щодня. Це допомагає не лише краще засвоювати матеріал, а й одразу застосовувати його на практиці.

Зрозуміти та систематизувати ці потреби можна за допомогою аналізу користувацького досвіду, відгуків, тестування прототипів і порівняння з іншими продуктами. Отримані дані дозволяють визначити найбільш затребувані функції та уникнути помилок, пов’язаних із недооцінкою зручності чи ефективності запропонованих рішень.

1.3.1 Визначення цільової аудиторії

Цільова аудиторія гейміфікованих платформ для вивчення мов охоплює широке коло користувачів різного віку, соціального статусу та професійної зайнятості. Найбільш активними користувачами подібних продуктів є молодь, студенти, школярі та фахівці, які прагнуть вивчати мову у зручному та цікавому форматі. Їх приваблює можливість поєднувати навчання з розвагою.

Для значної частини користувачів важливо, щоб платформа давала змогу швидко розпочати навчання без необхідності проходити складні процедури. Простий та інтуїтивно зрозумілий інтерфейс, чітка структура занять і привабливий візуальний стиль стають вагомими чинниками у виборі освітнього застосунку. Саме тому продукти, які відповідають сучасним стандартам зручності, мають вищий рівень залученості.

Окрему групу складають користувачі, які прагнуть вивчити нову мову для подорожей, переїзду або роботи за кордоном. Для них важлива практичність навчального контенту, швидкий доступ до базової лексики та

тем, пов’язаних із повсякденними ситуаціями. Такі користувачі, як правило, орієнтовані на прикладну користь від платформи.

Також до цільової аудиторії належать користувачі, які вже вивчали мову раніше, але прагнуть відновити або поглибити знання. Для них важливо мати можливість вибору рівня складності та проходження тематичних блоків відповідно до власного досвіду. Такі користувачі очікують від платформи більшої гнучкості та персоналізації.

Додатково варто враховувати інтерес до мовного навчання серед батьків, які шукають освітні інструменти для дітей. У цьому випадку важливо забезпечити елементи гейміфікації, що відповідають віковим особливостям, а також простоту контролю за прогресом. Таким чином, платформа може мати декілька груп користувачів, кожна з яких має свої специфічні потреби.

1.3.2 Очікування користувачів від освітніх платформ

Користувачі очікують від мовних платформ не лише якісного контенту, а й комфортної взаємодії з системою. Вони прагнуть бачити зрозумілу структуру уроків, логічну послідовність матеріалів і доступні пояснення. Особливо цінується наявність прикладів з повсякденного життя, які легко застосовувати в реальному спілкуванні. Простота інтерфейсу та швидкий доступ до основних функцій значно підвищують задоволеність користувача.

Сучасний користувач цінує можливість налаштовувати досвід навчання під себе. До таких очікувань належать вибір мови інтерфейсу, можливість обрати тему або рівень складності, інструменти для відстеження прогресу та функції повторення. Важливим також є надання зворотного зв’язку в реальному часі, який допомагає закріплювати знання та виправляти помилки одразу.

Крім цього, користувачі очікують на елементи заохочення – нагороди, досягнення, рівні, що створюють відчуття гри та підтримують мотивацію. Вони сприймають платформу як не лише навчальний інструмент, а й простір для самореалізації та змагання. Це особливо актуально для молоді, яка звикла до взаємодії з ігровими або соціальними сервісами.

1.3.3 Проблеми користувацького досвіду в існуючих рішеннях

Однією з поширених проблем користувацького досвіду є швидка втрата інтересу до навчання через одноманітність матеріалу. Багато платформ пропонують схожі типи вправ або повторювану структуру занять, що знижує мотивацію до регулярного використання. Відсутність новизни та несподіваних елементів у навчальному процесі призводить до зниження залученості навіть серед початково зацікавлених користувачів.

Ще одним викликом є недостатній рівень адаптації контенту до індивідуальних потреб. Деякі платформи не враховують рівень підготовки, інтереси або цілі користувача, через що навчання може здаватися надто легким або навпаки – складним і фруструючим. Відсутність можливості обирати теми або змінювати маршрут навчання обмежує гнучкість і робить досвід менш персоналізованим.

1.4 Визначення ключових функцій та можливостей розроблюваної системи

Визначення функціональних можливостей системи є важливим етапом при розробці будь-якого цифрового освітнього продукту. У контексті мовного навчання це означає врахування основних потреб користувачів, таких як доступ до тематичних блоків, можливість відстеження прогресу та створення персонального навчального маршруту.

Правильне проектування функцій забезпечує зручність використання та ефективність навчання.

Гейміфікована платформа повинна містити елементи, які стимулюють інтерес до навчального процесу. Це можуть бути досягнення, бали, рівні, внутрішня валюта або інші механізми заохочення. Такі інструменти не тільки підвищують залученість, а й сприяють повторюваності дій, що є важливим для закріплення знань. Система має бути гнучкою, щоб дозволити користувачу навчатися у комфортному темпі.

Важливо також приділити увагу загальній архітектурі застосунку. Вона має забезпечувати логічну побудову екранів, швидкий доступ до потрібних розділів та стабільну роботу на різних пристроях. Система повинна легко масштабуватися у разі збільшення кількості користувачів або додавання нових мов. Це підвищує надійність продукту та дозволяє легко розвивати його в майбутньому.

Технічна реалізація повинна базуватись на сучасному стеку технологій, який забезпечує швидкодію, безпеку та інтеграцію з базами даних. Окрему увагу слід приділити збереженню навчального прогресу та можливості авторизації через популярні сервіси. Загальна структура системи має підтримувати як поточну функціональність, так і потенціал для її розширення.

1.4.1 Основна логіка застосунку

Основна логіка гейміфікованого застосунку для вивчення мов передбачає поділ навчального процесу на тематичні блоки, які складаються з тестових завдань. Кожен блок присвячений окремій мовній темі, наприклад, граматичній конструкції або групі лексичних одиниць. Користувач поступово проходить ці блоки, отримуючи за правильні відповіді винагороди у вигляді віртуальних балів або досягнень.

Навчальний контент реалізується у вигляді інтерактивних тестів із різними типами запитань. Це можуть бути як вибір правильного варіанту, так і завдання на встановлення відповідності, заповнення пропусків або вибір слів у контексті. Такий формат дозволяє поєднувати тренування мовних навичок із залученням елементів гри, що робить навчання більш динамічним та цікавим.

Застосунок також має включати систему авторизації та особистого кабінету, де користувач зможе переглядати власний прогрес, кількість пройдених тестів, зароблені винагороди та відкриті досягнення. Це створює ефект персоналізованого навчання і дозволяє краще контролювати результати. Особистий кабінет виконує роль центру управління взаємодією з платформою.

Крім того, логіка застосунку передбачає можливість блокування деяких тестів, які відкриваються лише після досягнення певних умов, наприклад, накопичення певної кількості балів або проходження попередніх тем. Це підтримує структуру прогресії та заохочує користувача повертатися до навчання, щоб розблокувати нові можливості. Такий підхід сприяє формуванню стабільного навчального ритму.

1.4.2 Механізми гейміфікації та досягнень

Механізми гейміфікації в освітніх платформах виконують функцію мотиваційного інструменту, який стимулює користувачів до регулярної взаємодії з контентом. У межах платформи для вивчення мов до таких механізмів належать бали за правильні відповіді, накопичувані зірки, внутрішня валюта, а також прогресивне відкриття нових тем. Це створює враження гри з рівнями та заохоченням за кожне досягнення.

Окрему роль відіграє система досягнень. Користувач отримує спеціальні відзнаки за проходження певної кількості тестів, завершення тем або виконання особливих умов. Такі досягнення не лише надають відчуття

прогресу, а й формують додаткову мотивацію повертатися до платформи, щоб здобути нові нагороди. Це ефективно працює як для дорослих, так і для молодших користувачів.

Важливим елементом гейміфікації є візуалізація прогресу. Графічне представлення досягнень, кількості балів, рівнів або відсотку завершених блоків дозволяє користувачу бачити результат своїх зусиль. Це не лише підвищує залучення, а й допомагає оцінити власний розвиток у навчанні. Чітка структура успіхів підвищує відчуття контролю та досягнень.

Крім внутрішніх мотиваційних засобів, у платформу може бути інтегрована таблиця лідерів, де користувач бачить своє місце серед інших. Це створює елемент змагання і дозволяє активним учням порівнювати свої результати. Соціальний компонент гейміфікації підсилює ефект залучення та формує спільноту навколо навчального процесу.

1.4.3 Вибір стеку технологій для реалізації

Для реалізації платформи обрано сучасний стек технологій, який забезпечує стабільну роботу, гнучкість розвитку та зручність підтримки. На стороні клієнта використовується React як одна з найпопулярніших бібліотек для створення швидких і адаптивних інтерфейсів. Навігація реалізована за допомогою react-router, а для стилізації використовується Joy UI. Це дозволяє створити простий та водночас функціональний інтерфейс.

На серверній частині використовується Nest.js – фреймворк, який базується на TypeScript і має модульну структуру. Він забезпечує організовану архітектуру коду, зручну обробку запитів та інтеграцію з базою даних. Для зберігання інформації про користувачів, тести, результати та прогрес застосовується Firestore – масштабована хмарна база даних від Google, яка дозволяє обробляти дані в реальному часі.

Розгортання платформи виконується в хмарному середовищі Google Cloud з використанням Docker-контейнерів. Такий підхід гарантує

стабільність, ізоляцію середовищ і можливість швидкого масштабування. Уся система розроблена з урахуванням безпечної авторизації, збереження прогресу та подальшого розширення функціоналу.

1.5 Постановка задачі

Сучасні тенденції у сфері цифрової освіти та активне використання гейміфікації у навчальних системах створюють передумови для розробки нових мовних платформ, що поєднують ефективність навчання з високим рівнем залученості користувача. В умовах високої конкуренції виникає потреба у створенні веб-застосунку, який би враховував потреби різних категорій користувачів, зберігаючи при цьому простоту, інтуїтивність і мотиваційну привабливість.

Основною задачею є розробка платформи, що реалізує навчання через тематичні тести з інтерактивними завданнями, системою винагород, прогресу та досягнень. Додаток має забезпечувати можливість авторизації, перемикання мов, ведення особистого кабінету з історією проходжень, накопичення віртуальних балів та відкриття нових блоків на основі ігрової активності.

Передбачається реалізація повного циклу проектування та впровадження системи: від аналізу предметної галузі і потреб користувачів до створення архітектури, вибору технологічного стеку, розробки інтерфейсу, логіки гейміфікації та розгортання застосунку у хмарному середовищі. Результатом має стати працюючий продукт, здатний забезпечити мотиваційне та зручне середовище для вивчення іноземних мов.

2 ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Формалізація вимог до гейміфікованого навчального середовища

Процес формалізації вимог до гейміфікованого навчального середовища є ключовим етапом проектування, який визначає функціональні та нефункціональні особливості системи, на основі яких відбувається подальша архітектурна реалізація. У межах цієї платформи основним завданням є створення ефективного, зручного та мотиваційного інструменту для вивчення іноземних мов, що поєднує тестові завдання з ігровими механіками. Таким чином, вимоги формуються з урахуванням очікувань користувачів, принципів сучасного UX-дизайну, технічних можливостей обраного стеку технологій, а також стандартів якості веб-застосунків.

До базових функціональних вимог належить наявність реєстрації та авторизації користувачів, ведення персонального профілю, можливість проходження тестових завдань з поділом на теми та мовні рівні, а також накопичення і відображення результатів. Важливою вимогою є реалізація механізмів гейміфікації: система балів, досягнень, рейтингів, розблокування нових розділів після виконання певних умов, а також візуалізація прогресу. Інтерфейс має бути інтуїтивно зрозумілим і підтримувати адаптивну верстку для різних типів пристроїв, зокрема комп’ютерів, планшетів і смартфонів.

Нефункціональні вимоги охоплюють питання продуктивності, надійності, масштабованості та безпеки. Платформа має стабільно працювати з великою кількістю одночасних користувачів, оперативно обробляти дії в інтерфейсі, забезпечувати збереження та цілісність даних у хмарній базі даних Firebase Firestore. Важливо також дотримуватись принципів конфіденційності та захищеності персональної інформації користувачів відповідно до стандартів, зокрема використання

аутентифікації через Firebase Authentication. Питання розширення функціональності у майбутньому також враховується через модульний підхід до проектування серверної логіки.

Окрему категорію становлять вимоги до візуальної привабливості й залучення. Платформа має мотивувати користувача повертатися до навчання, використовуючи позитивні підкріплення, анімації при досягненнях, графічне оформлення досягнень та динамічну зміну інтерфейсу залежно від прогресу. Інтерфейс повинен бути простим, але не примітивним, і враховувати принципи когнітивного навантаження, не перевантажуючи екран зайвими елементами. Користувач має зосереджуватись на виконанні завдань, тоді як інші елементи служать допоміжними й стимулюючими.

Сформульовані вимоги поділяються також за ролями користувачів. З боку студента або звичайного користувача вимоги зосереджені на можливості навчання, відстеження прогресу та мотивації через гру. Адміністративна частина передбачає базові інструменти керування контентом, статистикою проходжень, а також можливість розширення тем та оновлення тестів без втручання у код. Таке розмежування дозволяє проектувати систему гнучко та адаптивно, готуючи її до масштабування як у горизонтальному, так і у вертикальному вимірі.

Під час формалізації враховувались також типові сценарії поведінки користувача, зокрема початковий вхід у систему, вибір теми, проходження тесту, отримання оцінки, перегляд статистики та повернення для проходження наступного рівня. Такий підхід дозволив сформувати чітке бачення логіки роботи кожного компонента платформи й закласти основу для подальшого функціонального моделювання.

На основі сформульованих вимог буде побудовано логіку подальших підрозділів, зокрема функціональні схеми, архітектуру, опис структури бази даних та вибір відповідних технологічних рішень. Таким чином, формалізація вимог виконує не лише роль постановки задачі, а й забезпечує

методологічну базу для всього етапу розробки гейміфікованої освітньої платформи.

2.2 Побудова функціональної моделі поведінки користувача

Побудова функціональної моделі поведінки користувача є важливим етапом у процесі проектування платформи, оскільки саме через взаємодію з системою користувач реалізує навчальні цілі. Завдяки правильному моделюванню вдається виявити ключові точки контакту, сценарії використання та логічні переходи між екранами і модулями. Це дозволяє мінімізувати ризики плутанини, підвищити зручність навігації й забезпечити логічну послідовність дій. У гейміфікованому навчальному середовищі поведінка користувача поєднує як елементи традиційного використання освітнього ПЗ, так і динаміку ігрового процесу, що потребує окремої уваги до логіки прогресії, мотивації та зворотного зв’язку.

Функціональне моделювання дає змогу представити всі можливі шляхи взаємодії у вигляді сценаріїв, які описують логіку дій користувача від початку сесії до завершення певної активності. Такі сценарії охоплюють не лише базові дії, як-от реєстрацію чи проходження тесту, але й гейміфіковані процеси – отримання досягнень, перегляд таблиці лідерів, зміну мови навчання тощо. Вони є основою для побудови діаграм послідовності, які деталізують внутрішні виклики до служб і бази даних, дозволяючи уточнити архітектуру та інтерфейси між компонентами.

У рамках цієї роботи визначено п’ять основних сценаріїв поведінки користувача: реєстрація або вхід до системи, проходження тесту з можливістю купівлі, отримання досягнення за результатами активності, зміна мови навчання та перегляд рейтингової таблиці. Кожен з них реалізований як окремий процес зі своєю логікою, умовами та результатами. Їх аналіз дає змогу виявити критичні точки взаємодії, що потребують

оптимізації, а також забезпечує надійне підґрунтя для реалізації відповідних компонентів системи.

Сценарій реєстрації або входу користувача є базовим етапом, що відкриває взаємодію з системою. Далі можна побачити відповідну діаграму послідовності (рисунок 2.1), яка демонструє усі основні етапи цього процесу. Користувач запускає застосунок, після чого має можливість обрати між створенням нового облікового запису або входом до вже існуючого. Обидві гілки реалізуються окремими послідовностями, які мають спільну кінцеву точку – перенаправлення до головного екрану платформи.

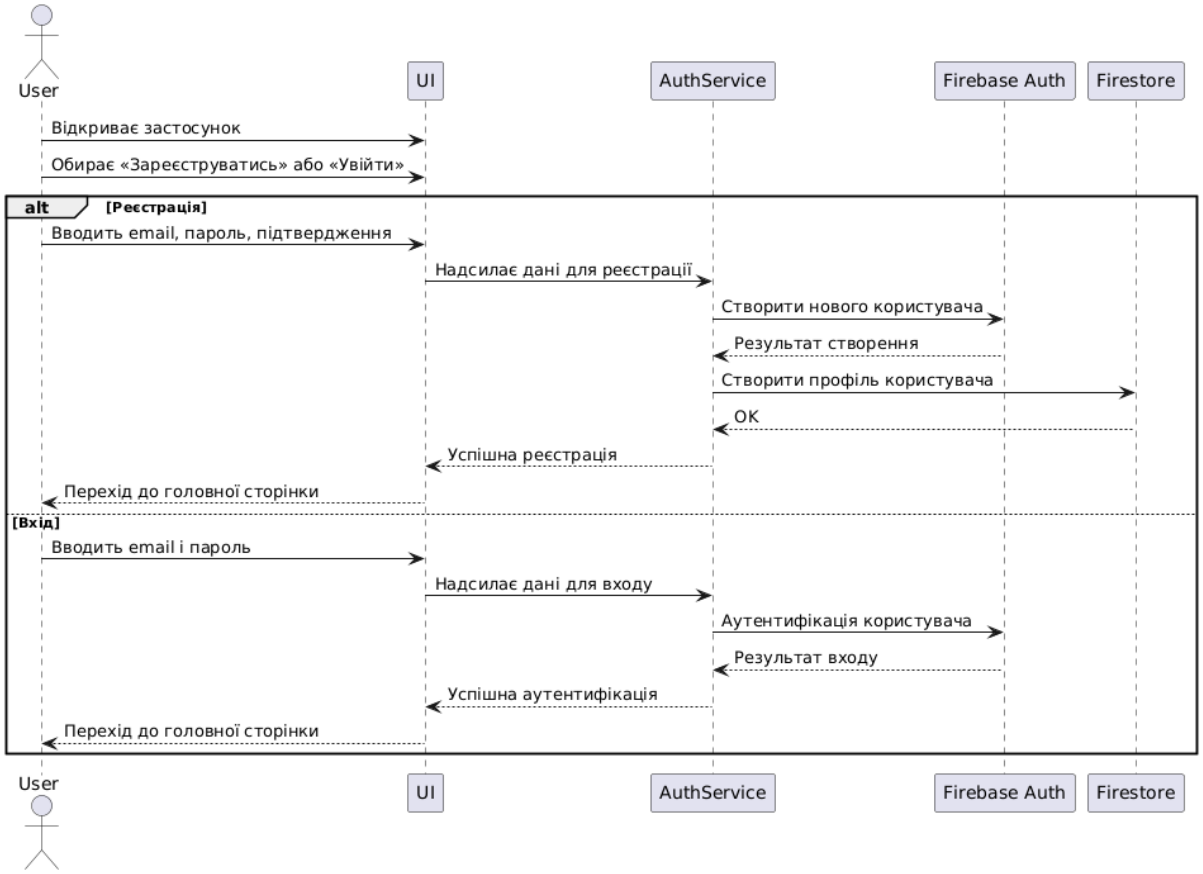

Рисунок 2.1 – Реєстрація чи аутентифікація користувача

У випадку реєстрації користувач вводить email, пароль та підтвердження паролю, після чого ці дані передаються до служби автентифікації. Вона взаємодіє з Firebase для створення нового облікового

запису, а після успішного створення додатково ініціюється запис профілю користувача в Firestore. Це дозволяє зберігати розширені дані, такі як мова, рівень чи прогрес, які не входять до базових атрибутів Firebase Authentication. Успішна реєстрація підтверджується клієнту, і користувач одразу отримує доступ до функціональності платформи.

У випадку входу користувач вводить свої облікові дані, які надсилаються до Firebase для перевірки автентичності. Після успішної перевірки застосунок отримує відповідь про успішну аутентифікацію, і користувач також переходить на головну сторінку. Завдяки цій діаграмі можна прослідкувати не лише взаємодію з UI, а й взаємодію між бекенд-сервісами, що забезпечують надійність і безпеку при доступі до освітньої платформи.

Процес проходження тесту в системі є центральним сценарієм взаємодії користувача з платформою. Він охоплює кілька етапів – від вибору теми й конкретного тесту до завершення перевірки знань та отримання оцінки. У цьому сценарії важливо не лише забезпечити безперебійну логіку виконання дій, а й реалізувати механізм доступу до платного контенту, що потребує окремого підтвердження перед початком тестування. Система повинна також підтримувати гнучкий цикл завдань, зберігати результати та одразу надавати зворотний зв’язок.

Далі можна побачити діаграму послідовності, яка ілюструє увесь цей сценарій із зазначенням задіяних сервісів, таких як TestService, PaymentService та Firestore (рисунок 2.2). Користувач розпочинає взаємодію з вибору теми, після чого отримує перелік доступних тестів, збережених у базі даних. Після відображення списку, користувач обирає конкретний тест, і система перевіряє його доступність. Якщо виявляється, що тест недоступний, ініціюється запит на покупку. Після підтвердження покупки користувач отримує дозвіл на проходження обраного тесту.

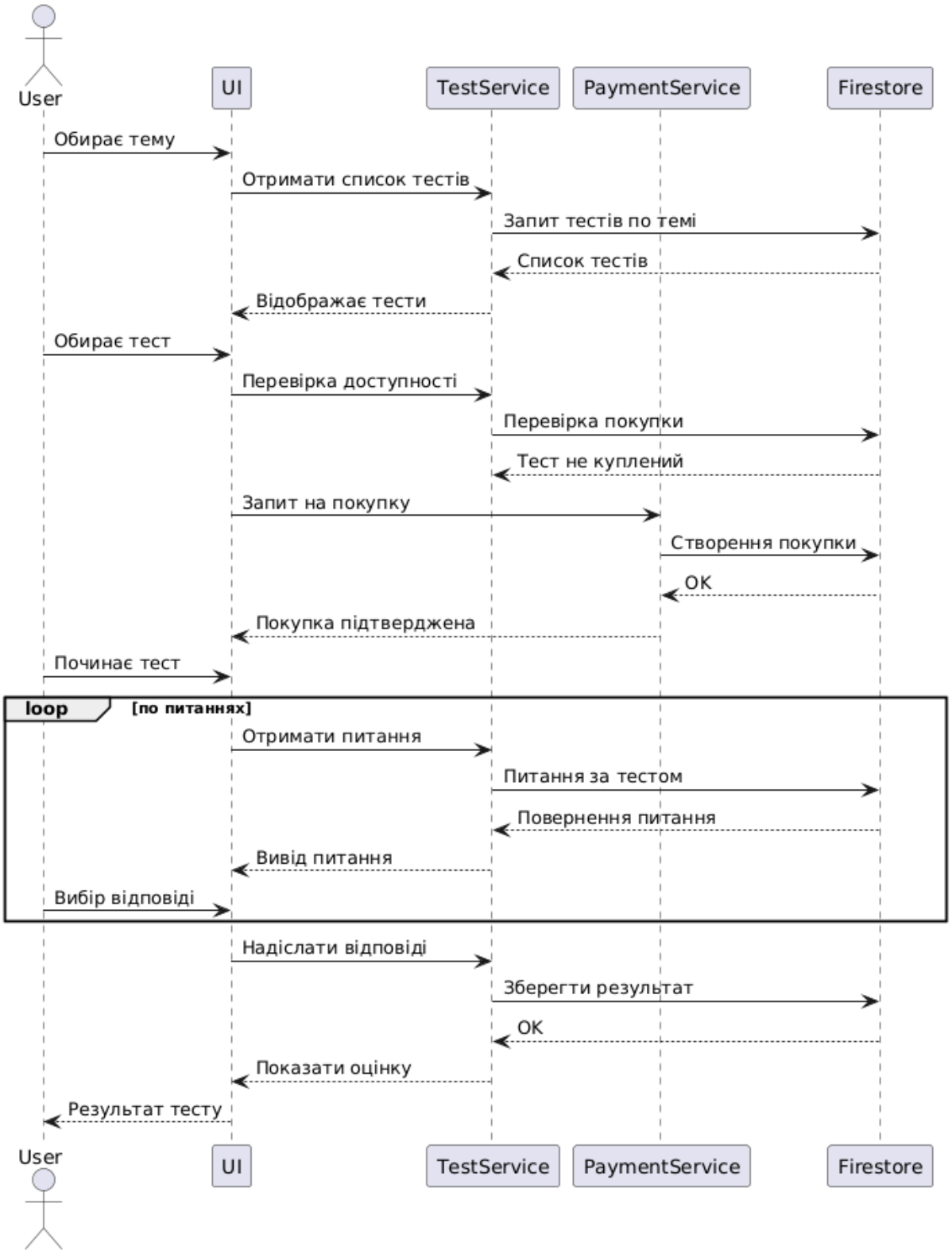

Рисунок 2.2 – Проходження тесту

Саме проходження тесту реалізується як цикл, у якому кожне питання витягується з бази даних окремо та передається на інтерфейс для

відображення. Користувач обирає відповідь, і вона одразу надсилається назад у систему. Такий підхід дозволяє будувати адаптивне тестування, де можна реалізувати перевірку на рівні кожного питання, включаючи логіку випадкового порядку, контроль часу чи адаптацію складності залежно від попередніх відповідей.

Після завершення останнього питання формується підсумковий результат, який зберігається в базі даних. Це дає змогу формувати статистику користувача, виводити аналітику у вигляді графіків або таблиць, а також реалізувати подальшу логіку нарахування балів або відкриття нових рівнів. Така структура забезпечує високу гнучкість та масштабованість у додаванні нових типів тестів, мов чи рівнів складності.

Сценарій отримання досягнень (рисунок 2.3) є важливою частиною гейміфікаційної складової платформи. Його мета – мотивувати користувача до регулярної взаємодії з системою шляхом надання нагород за досягнення певних умов, наприклад, завершення п’яти тестів. Досягнення реалізуються як окрема логіка, яка реагує на завершення певної активності, перевіряє, чи виконано задану умову, і при позитивному результаті генерує винагороду у вигляді зірок або алмазів.

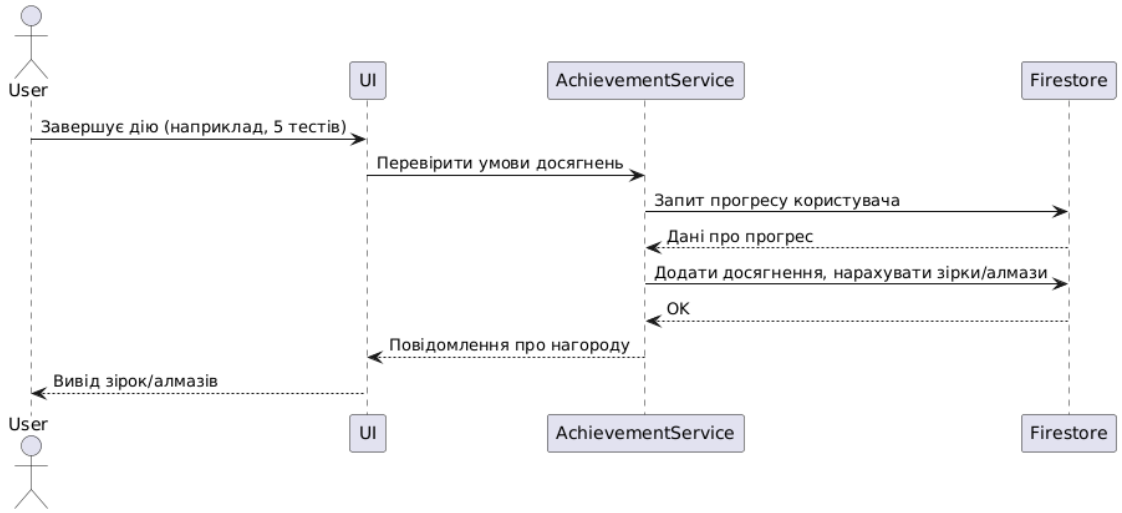

Рисунок 2.3 – Отримання досягнення

Коли користувач завершує дію, що потенційно може призвести до отримання досягнення, інтерфейс ініціює запит до сервісу досягнень. Цей сервіс звертається до бази даних для перевірки актуального прогресу користувача, порівнює його зі списком умов, і якщо досягнення справді виконано, додає відповідний запис у колекцію досягнень користувача. Паралельно нараховуються ігрові ресурси – зірки чи алмази – які можуть бути використані у майбутньому для відкриття нових рівнів чи тем.

Після завершення процесу користувач отримує повідомлення про нову нагороду через інтерфейс. Це повідомлення може супроводжуватися анімацією, спеціальним візуальним ефектом або звуком, що додатково посилює мотиваційний ефект. Такий сценарій є універсальним і може масштабуватись – список досягнень і відповідних умов може розширюватись без зміни логіки основної системи, а винагороди можуть адаптуватись до нових гейміфікаційних механік.

Сценарій зміни мови навчання є важливою частиною персоналізації платформи, що дозволяє користувачу адаптувати середовище до власних цілей і рівня володіння мовами. Можливість перемикання мови дає змогу не лише вивчати іншу мову, а й змінювати інтерфейс відповідно до нових потреб, що є особливо важливим для мультимовних користувачів або в контексті зміни навчального фокусу. Така гнучкість підвищує цінність платформи як універсального засобу для вивчення мов.

Далі можна побачити діаграму послідовності, яка ілюструє логіку зміни мови через налаштування профілю (рисунок 2.4). Користувач відкриває відповідний розділ у меню налаштувань, після чого обирає нову мову. Ця дія ініціює запит до сервісу користувача, який відповідає за обслуговування профілю. У рамках цього запиту змінюється значення мовного параметра у базі даних.

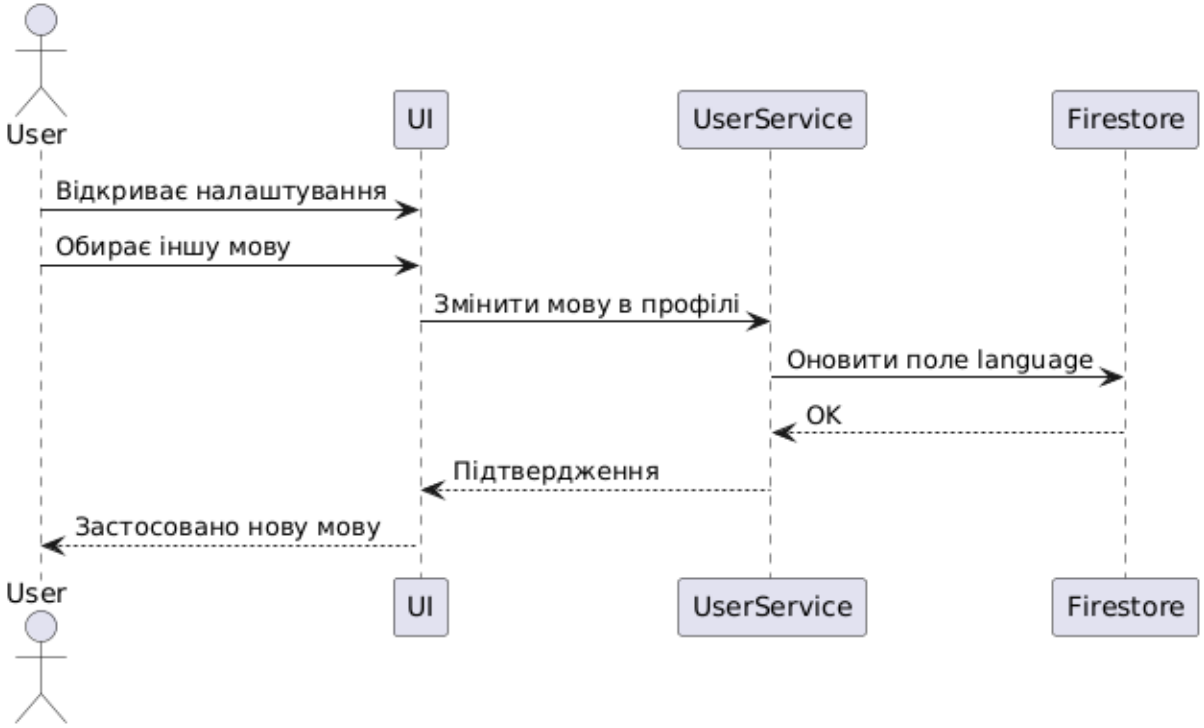

Рисунок 2.4 – Зміна мови навчання

Сервіс користувача надсилає запит до Firestore для оновлення відповідного поля language у профілі користувача. Після успішного завершення цієї операції система отримує підтвердження і передає його назад до інтерфейсу, який повідомляє користувача про застосування нових мовних налаштувань. Надалі це впливає на контент, який відображається в інтерфейсі, а також на тематику тестів та досягнень, адаптуючи їх під нову мовну ціль.

Завдяки такій реалізації забезпечується збереження персоналізованих налаштувань на стороні сервера, що дозволяє користувачу повертатися до навчання з різних пристроїв без повторного налаштування. Крім того, це створює передумови для реалізації багатомовної підтримки інтерфейсу в майбутньому, якщо платформа буде адаптована під користувачів з різних регіонів. Такий сценарій формує позитивний користувацький досвід та підтримує гнучкість платформи.

Функція перегляду таблиці лідерів є важливим елементом гейміфікації, яка дозволяє користувачам бачити свій прогрес у порівнянні з іншими. Вона виконує не лише інформаційну функцію, а й формує додаткову мотивацію за рахунок соціального змагання. Користувач може оцінити власні досягнення, відчути задоволення від потрапляння у верхні позиції або, навпаки, побачити необхідність активнішого навчання для покращення результатів. Особливо ефективним є поєднання лідерборду з системою винагород, що підсилює інтерес до регулярного використання платформи.

Далі можна побачити діаграму послідовності, яка демонструє процес взаємодії користувача з таблицею лідерів (рисунок 2.5). При відкритті відповідного розділу інтерфейс надсилає запит до сервісу лідерборду, який виконує запит до Firestore для отримання списку топових користувачів. Дані, що надходять з бази, включають загальний рейтинг, кількість набраних балів або зірок, а також ім’я або нікнейм користувача. Це забезпечує повноцінне уявлення про позицію кожного гравця у системі.

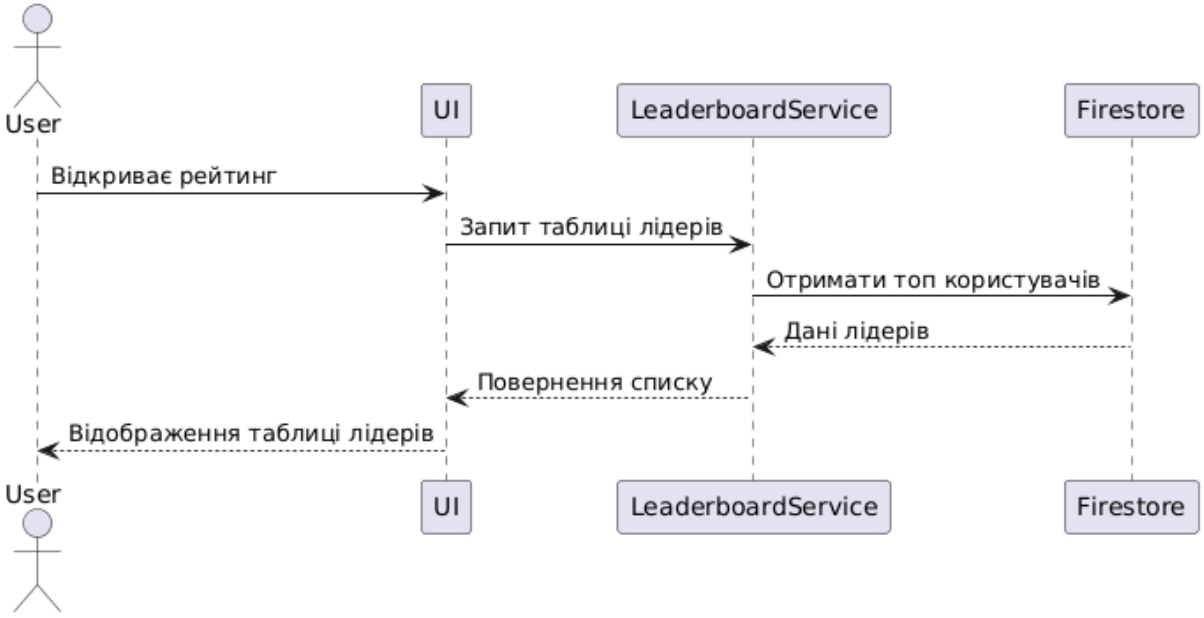

Рисунок 2.5 – Перегляд таблиці лідерів

Отриманий список передається назад до інтерфейсу, де він форматовано відображається користувачу. Таблиця може включати не лише позицію в загальному рейтингу, а й фільтри, наприклад, за друзями, країнами чи часовими періодами. Такий підхід дозволяє персоналізувати досвід перегляду й підвищує рівень залучення, оскільки користувач бачить не абстрактний список, а контекстуально значущі дані.

Реалізація цього сценарію ґрунтується на простій, але ефективній логіці взаємодії з базою даних, яка забезпечує актуальність інформації без зайвого навантаження на клієнтську частину. Таблиця лідерів може оновлюватися як за запитом користувача, так і автоматично з певною періодичністю. Такий механізм є ключовим для підтримання змагального духу та сталого інтересу до проходження тестів і здобуття нових досягнень.

2.3 Архітектурна концепція платформи мовного навчання

Архітектура гейміфікованої платформи для вивчення мов базується на принципах модульності, масштабованості та відокремлення відповідальностей. Основна ідея полягає у розділенні системи на незалежні компоненти, кожен з яких виконує окрему функціональну роль. Такий підхід дозволяє легше підтримувати і розвивати систему, а також забезпечує гнучкість у впровадженні змін без ризику порушення загальної стабільності.

Далі можна побачити архітектурну схему, яка ілюструє основні модулі та їх взаємозв’язки у межах запропонованої системи (рисунок 2.3). У центрі взаємодії розташовано API Gateway – посередницький компонент, який приймає HTTP-запити від клієнта і маршрутизує їх до відповідних бекенд-сервісів. Така структура спрощує керування трафіком, дозволяє реалізувати централізовану автентифікацію, логування та обмеження доступу.

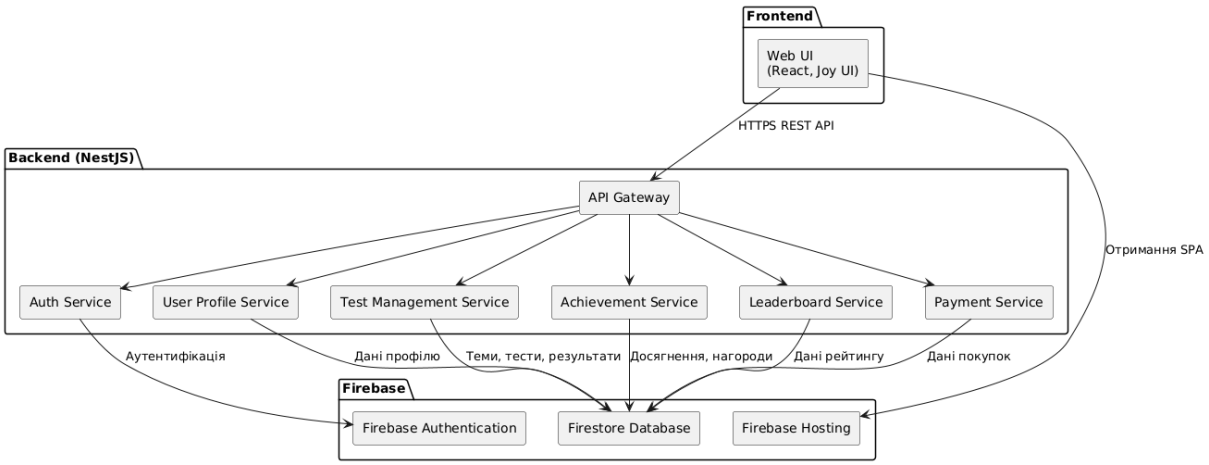

Рисунок 2.6 – Компонента архітектура платформи

Фронтенд платформи реалізовано як односторінковий додаток (SPA) на базі React із використанням бібліотеки Joy UI. Він відповідає за відображення інтерфейсу, взаємодію з користувачем і обмін даними з серверною частиною через REST API. Фронтенд отримує дані з API Gateway, а також завантажується через Firebase Hosting, що забезпечує швидке і надійне розгортання веб-інтерфейсу.

Серверна частина написана з використанням фреймворку NestJS і складається з кількох мікросервісів: сервіс автентифікації, сервіс профілю користувача, сервіс управління тестами, сервіс досягнень, сервіс рейтингу та сервіс оплати. Кожен із них обробляє запити у своїй предметній області й взаємодіє з базою даних Firestore. Наприклад, сервіс Test Management відповідає за зберігання і видачу тестів, обробку відповідей і збереження результатів.

Firebase виступає як хмарна платформа, яка інтегрує у собі засоби аутентифікації (Firebase Authentication), базу даних (Firestore Database) та хостинг (Firebase Hosting). Такий вибір зумовлений простотою інтеграції з фронтендом, високою продуктивністю у реальному часі та гнучкими можливостями масштабування. Дані користувачів, прогрес, досягнення та

рейтинг зберігаються централізовано, що забезпечує їх доступність з різних пристроїв.

Подібна архітектура дає змогу розгортати окремі компоненти незалежно один від одного, масштабувати сервіси залежно від навантаження, а також реалізовувати CI/CD-підхід до розробки та тестування. Це особливо важливо для освітніх платформ, які активно розвиваються, потребують частих оновлень функціоналу та безперебійної роботи у періоди високої активності.

2.4 Організація зберігання даних у Firebase Firestore

Для зберігання даних у гейміфікованій платформі для вивчення мов обрано Firebase Firestore – сучасну хмарну базу даних NoSQL, яка забезпечує зручну структуру зберігання, високу швидкість доступу, масштабованість і підтримку реального часу. Firestore ідеально підходить для застосунків, де дані часто оновлюються, потребують миттєвого відображення в інтерфейсі та синхронізації між користувачами. Його гнучка модель зберігання дозволяє зберігати дані у вигляді колекцій та документів, що значно спрощує структуру й адаптацію під майбутні зміни функціональності платформи.

У структурі Firestore кожна колекція є логічним об'єднанням документів певного типу. Наприклад, колекція users містить документи з інформацією про кожного користувача, включаючи email, обрану мову, поточний рівень, кількість зірок та алмазів, отримані досягнення та інші персоналізовані параметри. Колекція tests містить усі тести з їх метаданими, такими як назва теми, складність, список запитань і правильні відповіді. Це дозволяє швидко знаходити, фільтрувати та оновлювати тести за різними критеріями, не навантажуючи систему зайвими запитами.

Особливе місце в структурі бази займають колекції achievements та leaderboards. Колекція achievements містить шаблони всіх можливих

досягнень, а також окремі підколекції в documents користувачів з позначкою про досягнуті результати. Такий підхід дозволяє легко додавати нові досягнення без необхідності змінювати основну логіку додатку. Аналогічно, колекція leaderboards містить агреговані показники за останні періоди (тиждень, місяць), які оновлюються системними процесами та використовуються для формування таблиці лідерів. Оновлення може виконуватись через cron-джоби або події в реальному часі.

Технічна реалізація зберігання у Firestore також включає підтримку правил безпеки, які визначають, хто і за яких умов може читати або змінювати конкретні колекції та поля. Наприклад, користувач може змінювати лише свій профіль, бачити загальні тести, але не мати доступу до чужих результатів. Це реалізується за допомогою Firebase Security Rules, які перевіряють автентифікацію та відповідність UID користувача. Таке обмеження особливо важливе у контексті захисту персональних даних і підтримання довіри користувачів до платформи.

Firestore також надає можливості для ефективного кешування, офлайн-доступу та зменшення кількості читань, що особливо важливо для мобільних користувачів або при нестабільному з’єднанні. Кожен запит може бути оптимізований через індекси, а складні запити підтримуються за допомогою комбінованих умов фільтрації. Наприклад, можна запитати всі тести з теми «Їжа», рівнем «A2» і доступні для поточного користувача. Це дозволяє будувати інтелектуальну логіку видачі завдань без складної серверної обробки.

Окремої уваги потребує зберігання результатів тестування. Кожен прохід тесту зберігається як окремий документ у колекції testResults з прив’язкою до користувача, тесту, часу проходження та кількості правильних відповідей. На основі цих даних система може формувати аналітику, відстежувати прогрес, генерувати рекомендації та виявляти закономірності. Така модель є масштабованою і дозволяє в майбутньому реалізувати адаптивне навчання або рекомендаційні алгоритми.

Таким чином, структура бази даних у Firestore відповідає логіці застосунку, є розширюваною, безпечною та гнучкою. Її проектування базується на принципах розділення даних за логічними сутностями, підтримки швидкого доступу та інтеграції з іншими сервісами Firebase. Це забезпечує надійну основу для подальшого розвитку гейміфікованої платформи, включаючи додавання нових функцій, масштабування під більшу кількість користувачів і підтримку складніших сценаріїв навчання.

2.5 Технологічне підґрунтя реалізації програмної системи

Технологічне підґрунтя реалізації гейміфікованої платформи для вивчення мов охоплює сукупність інструментів, фреймворків, мов програмування та середовищ, які забезпечують ефективну розробку, розгортання і підтримку всіх компонентів системи. Основний акцент зроблено на сучасних веб-технологіях, які дозволяють створити інтерактивний, адаптивний і масштабований застосунок із підтримкою гейміфікації, персоналізації та швидкої взаємодії з користувачем у реальному часі. Вибір стеку технологій базується на принципах модульності, повторного використання коду, безпеки, простоти інтеграції з хмарними сервісами та підтримки CI/CD.

Фронтенд-платформа побудована на базі React – популярної JavaScript-бібліотеки для створення інтерфейсів, розробленої компанією Meta. Вона забезпечує високу швидкість рендерингу, гнучкість компонування та потужні засоби керування станом. Для візуального оформлення застосовано UI-бібліотеку Joy UI від MUI, яка дозволяє реалізовувати сучасні адаптивні інтерфейси з підтримкою тем, анімацій і кастомізації без значних витрат на розробку. Взаємодія з серверною частиною реалізується через HTTP-запити до REST API, які підтримують авторизацію, обмін даними та керування сесіями.

Для роботи з формами, валідацією даних і керуванням подіями у клієнтській частині використовується бібліотека React Hook Form, яка дозволяє зменшити обсяг коду, підвищити продуктивність і забезпечити контроль над усіма полями форми. Для маршрутизації в односторінковому застосунку використовується React Router, що дозволяє реалізувати логічну навігацію між сторінками без перезавантаження. Уся логіка фронтенду зібрана та збирається у вигляді SPA за допомогою Vite – високошвидкісного інструменту для розробки та білду JavaScript-додатків, який забезпечує гаряче оновлення модулів, мінімізацію коду і високу швидкість старту.

Бекенд платформи реалізований з використанням NestJS – прогресивного Node.js фреймворку, який забезпечує модульну архітектуру, підтримку декораторів, інтеграцію з TypeScript і відповідність принципам SOLID. NestJS дозволяє створювати масштабовані серверні додатки, що легко підтримуються і розширюються. Серверна частина організована як набір мікросервісів, кожен з яких виконує окрему функцію: автентифікація, облік користувачів, управління тестами, система досягнень, оплата та рейтинг. Обмін даними між сервісами може відбуватись через HTTP або через повідомлення за допомогою брокерів, якщо система буде масштабована у майбутньому.

Для зберігання даних використовується Firebase Firestore – хмарна база даних NoSQL з підтримкою реального часу. Вона інтегрується як з фронтендом, так і з бекендом і дозволяє зберігати дані у вигляді колекцій та документів. Завдяки автоматичній синхронізації даних, вона ідеально підходить для збереження профілів користувачів, результатів тестів, досягнень і таблиць лідерів. Інтеграція з Firebase Authentication дозволяє реалізувати безпечну автентифікацію з перевіркою email та підтримкою безпарольного входу або Auth-ідентифікації, якщо це буде потрібно.

Для хостингу застосунку використовується Firebase Hosting – швидке й надійне рішення для розгортання статичних веб-додатків, яке підтримує HTTPS, автоматичне масштабування, кешування і можливість підключення

власного домену. Це дозволяє розгорнути SPA без необхідності налаштовувати окремий сервер. У випадку, якщо система розшириться, її окремі компоненти можуть бути перенесені на інші хмарні платформи або контейнеризовані через Docker.

Розробка системи підтримується через використання GitHub для керування версіями, а також реалізацію CI/CD-процесів із автоматичним білдом, тестуванням і деплоєм. Це дозволяє скоротити час між розробкою нової функції та її впровадженням, а також забезпечити стабільну якість коду. Тестування фронтенду здійснюється за допомогою Jest та React Testing Library, тоді як для бекенду використовуються можливості NestJS Test Framework.

Таким чином, обраний стек технологій відповідає сучасним вимогам до веб-розробки, дозволяє забезпечити швидку реакцію інтерфейсу, масштабовану логіку серверної частини, захищене зберігання даних і просте розгортання. Усі компоненти інтегруються між собою на основі відкритих стандартів і добре підтримуються спільнотою, що дозволяє ефективно вирішувати технічні завдання на кожному етапі життєвого циклу програмної системи.

2.6 Визначення критеріїв якості та показників ефективності системи

Оцінювання якості програмної системи є ключовим етапом при завершенні етапу проектування та подальшій реалізації. Воно дозволяє забезпечити відповідність розробленого продукту вимогам користувача, виявити слабкі місця, підвищити надійність і гарантувати ефективне функціонування в різних умовах експлуатації. У контексті гейміфікованої платформи для вивчення мов особлива увага приділяється не лише технічним показникам, а й таким характеристикам, як зручність інтерфейсу, швидкість відповіді, мотиваційна ефективність і адаптивність до потреб

користувача. Для оцінки використовуються як кількісні, так і якісні критерії, що охоплюють повний спектр вимог до сучасного вебзастосунку.

Першим критерієм виступає функціональна повнота. Система має підтримувати всі заявлені функції – реєстрацію та автентифікацію користувача, вибір і проходження тестів, накопичення досягнень, зміну мови, перегляд таблиці лідерів. Усі ці функції реалізуються з урахуванням логіки сценаріїв взаємодії, описаних на етапі функціонального моделювання. Для перевірки цього критерію використовуються модульні тести, ручне тестування UI та автоматизовані end-to-end тести. Усі компоненти мають реагувати передбачувано, незалежно від послідовності дій користувача або стану його сесії.

Другий важливий критерій – продуктивність. Застосунок має швидко реагувати на дії користувача, забезпечувати мінімальну затримку при завантаженні сторінок і обробці запитів. Для цього проводиться вимірювання часу відгуку серверних запитів, оцінка часу рендерингу компонентів інтерфейсу, а також використовується інструменти Lighthouse або Web Vitals для виявлення можливих вузьких місць. Firestore із доступом у реальному часі дозволяє мінімізувати затримки при оновленні даних, а використання Vite на фронтенді прискорює завантаження додатку. Загальна ціль – забезпечити час відгуку менше ніж 100 мілісекунд для найбільш критичних дій користувача.

Наступним критерієм є зручність використання, або usability. Інтерфейс має бути інтуїтивно зрозумілим, візуально привабливим і адаптованим для різних пристроїв. Для цього дизайн платформи реалізований із використанням Joy UI, що забезпечує адаптивну верстку та доступність. Логіка переходів між екранами відповідає сценаріям навчання, а взаємодія з платформою не викликає ускладнень навіть у користувача без технічної підготовки. Оцінка цього критерію може проводитись шляхом юзабіліті-тестування з реальними користувачами, збору зворотного зв’язку,

анкетування та аналізу поведінкових метрик (час на завдання, кількість кліків, повернень назад).

Іншим важливим критерієм є масштабованість і розширюваність. Платформа має легко адаптуватись до зростання кількості користувачів, додавання нових мов, тем, тестів або функціональних модулів. Мікросервісна архітектура NestJS дозволяє незалежно розгортати і оновлювати компоненти системи, а використання Firestore забезпечує горизонтальне масштабування без втрати продуктивності. Цей критерій особливо важливий у контексті можливого розширення платформи до повноцінного освітнього середовища з підтримкою курсів, груп, повідомлень та індивідуальних рекомендацій.

Окрему групу становлять показники безпеки та стійкості до помилок. Вони передбачають захист персональних даних, запобігання несанкціонованому доступу, збереження цілісності інформації та стійке поводження у випадку збоїв. Firebase Authentication надає вбудовану підтримку захисту даних користувача, включаючи обмеження доступу, перевірку пошти, шифрування, логування спроб входу. Крім того, безпека бази даних додатково забезпечується через правила доступу Firestore, які обмежують операції згідно з роллю або UID користувача. У випадку мережевих помилок, дані кешуються на клієнті, а відправка повторюється при відновленні з’єднання.

Загальне оцінювання якості програмної системи завершується визначенням відповідності її цільовому призначенню. У випадку гейміфікованої платформи це означає, що система має не просто надавати навчальні матеріали, а й стимулювати користувача до регулярної активності, створювати позитивний досвід, підтримувати змагальність і розвиток. Це може бути оцінено на основі кількості завершених тестів, частоти повернень у додаток, часу, проведеного в системі, кількості нарахованих досягнень.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Інструменти реалізації платформи

Реалізація програмної платформи для вивчення мов з гейміфікованими елементами потребує чіткого визначення інструментів, які забезпечують ефективну розробку, масштабованість і зручність у підтримці системи. При виборі технологічного стеку враховувались сучасні тенденції у веброзробці, а також вимоги до продуктивності, безпеки, адаптивності інтерфейсу та інтеграції з хмарними сервісами. Компоненти системи було поділено на клієнтську та серверну частини, кожна з яких реалізується із застосуванням відповідних інструментів та фреймворків.

Клієнтська частина була реалізована з використанням сучасних JavaScript-бібліотек та засобів для створення односторінкових застосунків. Особливу увагу приділено компонентній структурі, зручності стилізації, маршрутизації та керуванню станом інтерфейсу. Інструменти було обрано з урахуванням їхньої підтримки адаптивної верстки, можливості розширення і сумісності з бекендом.

Серверна частина реалізована з використанням потужного серверного фреймворку, що базується на TypeScript і дозволяє дотримуватись принципів модульного проектування. Для зберігання та обміну даними обрано хмарну базу NoSQL, яка зручно інтегрується з іншими сервісами, зокрема автентифікацією та зберіганням файлів. Деталізований опис кожного інструменту, його функціонального призначення та ролі у системі представлено в наступних підрозділах.

3.1.1 Вибір мови програмування та супутніх бібліотек

Для реалізації як клієнтської, так і серверної частини гейміфікованої платформи було обрано мову програмування TypeScript. Вона є

надмножиною JavaScript і дозволяє визначати типи змінних, параметрів, об’єктів і функцій, що підвищує надійність і передбачуваність коду. Завдяки цьому зменшується кількість помилок на етапі розробки та спрощується підтримка коду в довгостроковій перспективі. TypeScript має широку підтримку у середовищі веброзробки й сумісний із усіма основними бібліотеками та фреймворками.

У клієнтській частині TypeScript поєднується з декларативним стилем програмування React. Це дозволяє чітко описувати, як саме має виглядати інтерфейс у певному стані застосунку. Оголошення типів у компонентах, станах, пропсах і ефектах дає можливість швидко знаходити помилки ще до виконання програми, що особливо важливо при великій кількості взаємодіючих елементів. Завдяки цьому забезпечується гнучкість у розробці складних UI-компонентів без втрати керованості.

Застосування типізації на боці сервера, побудованого на Nest.js, дозволяє ефективно моделювати сутності, DTO-об'єкти, запити і відповіді, а також контрактні типи для взаємодії між модулями. Це створює передумови для побудови надійної архітектури із суворою логікою контролю відповідності даних. У разі зміни структури запиту або відповіді TypeScript миттєво сигналізує про невідповідність, що значно прискорює процес рефакторингу.

Ще однією перевагою використання TypeScript є підтримка сучасних можливостей ECMAScript, таких як асинхронні функції, деструктуризація, об'єктна модель і класи. Це дозволяє розробляти код згідно з сучасними стандартами без потреби в додатковій компіляції чи налаштуваннях. TypeScript також має потужні засоби автодоповнення, інтеграцію з IDE та систему linting-аналізу, що позитивно впливає на якість коду в командній розробці.

Крім самої мови, у межах цього проекту активно використовуються базові супутні бібліотеки, такі як axios для HTTP-запитів, classnames для гнучкої генерації класів, date-fns для роботи з датами та інші утилітарні

пакети. Усі вони типізовані або мають підтримку TypeScript, що дозволяє зберігати консистентність у коді та уникати помилок, пов’язаних із неправильним форматом аргументів чи поверненням некоректних значень.

Вибір TypeScript і типізованого оточення для реалізації повністю відповідає вимогам до проєкту, що поєднує в собі інтерактивну взаємодію, збереження даних, обробку сесій та реалізацію гейміфікаційних механізмів. Завдяки типізації забезпечується висока стабільність платформи, що дозволяє безпечно додавати нові модулі, розширювати функціонал та масштабувати проєкт без втрати контролю над структурою коду.

3.1.2 Інструменти розробки клієнтської частини

Клієнтська частина платформи реалізована як односторінковий застосунок з використанням бібліотеки React. Такий підхід дозволяє формувати інтерфейс динамічно без необхідності перезавантаження сторінки, що забезпечує швидку реакцію на дії користувача. React працює на основі концепції віртуального DOM, завдяки чому досягається оптимізація оновлення інтерфейсу, зменшення кількості зайвих рендерів і підвищення продуктивності застосунку. Компонентний підхід дозволяє розбивати функціональність на незалежні блоки, кожен з яких виконує окрему роль в інтерфейсі.

Для стилізації і побудови візуального оформлення було обрано бібліотеку Joy UI, що є частиною екосистеми MUI. Вона забезпечує гнучке кастомізоване оформлення компонентів, підтримку тем, адаптивну верстку та вбудовану типізацію. Це дозволяє створювати інтерфейс, який відповідає сучасним вимогам до дизайну, забезпечуючи при цьому зручність користування на різних пристроях. Компоненти Joy UI легко інтегруються з React і забезпечують повну підтримку accessibility-стандартів.

Організація маршрутизації в застосунку реалізована за допомогою бібліотеки React Router. Вона дозволяє налаштовувати переходи між

сторінками, включаючи як публічні маршрути, так і маршрути, що потребують автентифікації. Крім того, підтримується вкладена навігація, обробка параметрів у URL та захист маршруту на основі стану користувача. Це дає змогу реалізувати логічну структуру платформи: головна сторінка, профіль, список тестів, сторінка результатів, таблиця лідерів та інші.

Для керування формами і валідацією введених даних застосовується React Hook Form. Ця бібліотека мінімізує кількість рендерів, має зручну інтеграцію з кастомними компонентами та забезпечує просте підключення валідаційних правил. Вона активно використовується на сторінках авторизації, реєстрації, зміни налаштувань мови, проходження тесту та надсилання відповідей. Її типізоване API добре інтегрується з TypeScript, що дозволяє ефективно перевіряти коректність даних ще до їх відправки.

Збірка застосунку та локальний сервер розробки реалізовано за допомогою Vite – сучасного інструмента для високошвидкісної розробки проєктів на основі JavaScript і TypeScript. Vite забезпечує миттєве перезавантаження сторінки, гаряче оновлення модулів і мінімальні затримки при старті застосунку. У порівнянні з традиційними бандлерами, він дозволяє суттєво зекономити час на етапі розробки, зберігаючи при цьому високу якість і стабільність збірки у production.

Крім основних інструментів, у клієнтській частині використовуються додаткові утиліти та підходи: зберігання токенів у sessionStorage, підключення до серверного API з обробкою помилок, глобальні контексти для теми та мови, локалізація інтерфейсу. Усе це забезпечує не лише функціональність, а й зручність користування, що є ключовим фактором для підтримки довгострокового інтересу користувача до платформи.

3.1.3 Фреймворки та засоби серверної частини

Для реалізації серверної частини платформи було використано фреймворк Nest.js, побудований на базі Node.js та TypeScript. Nest.js є

сучасним рішенням для побудови надійних серверних застосунків, що підтримує архітектуру з чітким поділом на модулі, контролери та сервіси. Такий підхід дозволяє створювати масштабовані, зрозумілі й тестовані бекенд-системи, що повністю відповідає потребам платформи з великою кількістю взаємодіючих компонентів.

Архітектура сервера дотримується принципів розділення відповідальностей. Контролери відповідають за обробку HTTP-запитів і повернення відповідей, сервіси реалізують бізнес-логіку, а DTO-класи застосовуються для передачі типізованих даних. Це дозволяє легко відслідковувати потік даних між клієнтом і сервером, а також забезпечує передбачувану структуру коду. Кожна функціональна частина системи, така як управління користувачами, тестами, досягненнями чи оплатами, винесена в окремий модуль.

У Nest.js інтегровано систему декораторів, що забезпечує зручне керування маршрутами, валідацією вхідних даних, інжекцією залежностей та обробкою помилок. Завдяки цьому можна реалізовувати потужну серверну логіку з мінімальними зусиллями. Для обробки запитів на захищені маршрути використовується middleware-логіка, яка перевіряє токени доступу, передані з клієнтської частини через Auth0. Валідація здійснюється автоматично за допомогою класів і декораторів з бібліотеки class-validator.

Важливою особливістю Nest.js є підтримка інтерфейсів для роботи з різними типами баз даних. У даному випадку інтеграцію з Firestore реалізовано через Firebase SDK, що дозволяє зчитувати, створювати, оновлювати та видаляти документи у відповідних колекціях. Кожен сервіс Nest.js має відповідну обгортку для Firestore, що забезпечує типізовану взаємодію з базою даних. Це дозволяє зберігати результати тестів, дані користувачів, досягнення, таблиці лідерів та іншу інформацію у структурованому вигляді.

Серверна частина також обробляє логіку гейміфікаційної системи. У відповідних модулях реалізовано перевірку умов для досягнень, нарахування зірок та алмазів, оновлення профілю користувача та створення нових записів у Firestore. Усі ці операції виконуються із забезпеченням ідентефікації користувача, перевірки прав доступу та обробки можливих винятків. Таким чином, сервер не лише обробляє запити, а й підтримує внутрішню логіку навчального процесу.

Усі серверні модулі легко масштабуються, можуть оновлюватися незалежно один від одного, а також доповнюватися новими маршрутами або логікою без порушення стабільності системи. Така організація бекенду дозволяє швидко адаптувати платформу до нових вимог, додавати підтримку інших мов, рівнів складності або типів контенту. Завдяки використанню Nest.js як основного серверного фреймворку реалізація вийшла структурованою, зрозумілою і готовою до подальшого розвитку.

3.1.4 База даних Firestore

Для зберігання даних у платформі використовується сучасна база даних Firestore – хмарна NoSQL база даних, яка є частиною екосистеми Firebase. Вона дозволяє зберігати інформацію у форматі колекцій і документів, що забезпечує гнучкість у структурі та високу швидкість доступу до даних. Firestore має підтримку збереження в реальному часі, що дозволяє оперативно оновлювати інформацію на клієнті без перезавантаження сторінки, наприклад, у таблиці лідерів або при оновленні профілю користувача.

У межах платформи створено кілька основних колекцій: users, tests, achievements, testResults, purchases та leaderboard. Кожна колекція має унікальну структуру документів, адаптовану під логіку відповідного модуля. Збереження і читання даних реалізовано через Firebase SDK, що забезпечує безпосередню інтеграцію з бекендом і підтримує типізовану

роботу з об’єктами. Для обмеження доступу до даних використовуються Firestore Security Rules, які дозволяють читати або змінювати документи лише автентифікованим користувачам згідно з їх UID.

3.1.5 Сервіси для автентифікації та зберігання файлів

Для реалізації автентифікації в платформі використано сервіс Auth0, який забезпечує безпечний вхід користувача з можливістю розширеної конфігурації. Auth0 підтримує перевірку електронної пошти, управління сесіями, а також інтеграцію з іншими провайдерами ідентифікації, що дозволяє в майбутньому легко додати вхід через Google або Facebook. У клієнтській частині використовується офіційна бібліотека Auth0 для React, яка дозволяє відстежувати статус користувача, отримувати токени доступу та захищати маршрути в інтерфейсі.

Для зберігання файлів, таких як зображення профілю або інші ресурси, застосовується Google Cloud Storage (Bucket), який забезпечує надійне та масштабоване хмарне середовище для збереження статичних об’єктів. Завантаження та доступ до файлів реалізовано через підписані URL або через прямий доступ із Firebase SDK.

3.2 Реалізація серверної частини платформи

Для забезпечення взаємодії з клієнтським застосунком активується підтримка CORS з чітко визначеними дозволеними доменами. Крім того, глобально підключається механізм валідації вхідних даних, що дозволяє централізовано контролювати коректність запитів. Також відбувається ініціалізація підключення до Firebase через firebase-admin, де використовуються параметри, збережені у конфігураційному сервісі, – весь цей процес представлено в лістингу 3.1.

Лістинг 3.1 – Програмний код, що відповідає за ініціалізацію серверного застосунку Nest.js з підтримкою CORS, глобальною валідацією та інтеграцією з Firebase

```
async function bootstrap() {
  const app = await NestFactory.create(AppModule);
  app.enableCors({
    origin:                    ['http://localhost:5173',
'https://...run.app'],
    credentials: true,
  });
  app.useGlobalPipes(new ValidationPipe());
  const config = app.get(ConfigService);
  initializeApp({
    credential: cert({
      projectId: config.getOrThrow('FIREBASE_PROJECT_ID'),
      privateKey:
config.getOrThrow('FIREBASE_PRIVATE_KEY'),
      clientEmail:
config.getOrThrow('FIREBASE_CLIENT_EMAIL'),
    }),
    storageBucket:
`${config.getOrThrow('FIREBASE_PROJECT_ID')}.firebasestorage.a
pp`, });
  await app.listen(process.env.PORT ?? 3000);
}
bootstrap();
```

У гейміфікованій платформі важливою складовою є механізм мотивації користувача через систему досягнень. Для цього в коді визначено кілька типів нагород, кожен з яких відповідає певному критерію успішності. CompletedQuizzesAmountAward активується після досягнення певної кількості завершених тестів незалежно від мови. CompletedQuizzesAmountByLanguageAward враховує аналогічний критерій, але в межах окремої мови навчання. Також передбачені нагороди за

завершення тематичних блоків: CompletedQuizTopicsAmountAward та CompletedQuizTopicsAmountByLanguageAward. Усі чотири типи об’єднані в узагальнений тип Award, що дозволяє зручно оперувати всіма різновидами нагород в єдиному форматі – це відображено у лістингу 3.2.

Лістинг 3.2 – Програмний код, що описує типи гейміфікаційних нагород у системі на основі кількості завершених тестів, тем і мови навчання

```
export type CompletedQuizzesAmountAward = {
  id: string, title: string, threshold: number, score: number, reward: number, type: 'completedQuizzesAmount'
};

export type CompletedQuizzesAmountByLanguageAward = {
  id: string, title: string, threshold: number, language: LanguageCode, score: number, reward: number, type: 'completedQuizzesAmountByLanguage'
};
export type CompletedQuizTopicsAmountAward = {
  id: string, title: string, threshold: number, score: number, reward: number, type: 'completedQuizTopicsAmount'
};
export type CompletedQuizTopicsAmountByLanguageAward = {
  id: string, title: string, threshold: number, language: LanguageCode, score: number, reward: number, type: 'completedQuizTopicsAmountByLanguage'
};

export type Award =
  | CompletedQuizzesAmountAward
  | CompletedQuizzesAmountByLanguageAward
  | CompletedQuizTopicsAmountAward
  | CompletedQuizTopicsAmountByLanguageAward;
```

У наведеному фрагменті реалізовано функцію getAwards, яка відповідає за отримання даних про всі нагороди системи з хмарного сховища Firebase Storage. Для цього виконується звернення до JSON-файлу awards.json, зчитування його вмісту та десеріалізація у вигляді масиву об’єктів типу Award. Увесь механізм завантаження та обробки даних зображено в лістингу 3.3.

Лістинг 3.3 – Програмний код, що реалізує отримання масиву нагород із JSON-файлу, розміщеного в хмарному сховищі Firebase Storage

```
export const getAwards = async (): Promise<Award[]> => {
  const file = getStorage().bucket().file('awards.json');
  const contents = await file.download();
  return JSON.parse(contents.toString()) as Award[];
};
```

У наступному фрагменті коду визначено типи даних, що описують структуру тестового контенту платформи. Тип QuizTopic об’єднує окремі тести в тематичні блоки, які мають власний ідентифікатор, назву та іконку. Кожен тест типу Quiz містить набір питань, а також інформацію про нагороду, вартість і кількість балів. Типи Question та Answer деталізують кожне запитання тесту та його варіанти відповідей – загальна структура подана у лістингу 3.4.

Лістинг 3.4 – Програмний код, що визначає структуру типів для тем тестів, самих тестів, питань і відповідей у навчальній платформі

```
export type QuizTopic = {
  id: string, label: string, icon: string, quizzes: Quiz[]
};

export type Quiz = {
  id: string, label: string, score: number, price: number,
reward: number, questions: Question[]};
```

Продовження лістингу 3.4

```
export type Question = {
  label: string, answers: Answer[], correctId: string
};

export type Answer = {
  id: string, label: string
};
```

Цей фрагмент реалізує функцію getQuizzes, яка відповідає за завантаження всієї структури тестів із хмарного сховища. Джерелом даних є файл quizzes.json, що містить усі теми тестів, розділені за мовами. За допомогою API Firebase Storage виконується доступ до цього файлу, його зчитування та обробка. Вміст файлу десеріалізується у формат, де ключем є мовний код, а значенням – масив тем типу QuizTopic. Такий підхід дозволяє ефективно підтримувати мультимовність навчального контенту. Весь процес отримання та структурування даних подано у лістингу 3.5.

Лістинг 3.5 – Програмний код, що завантажує структуру вікторин з файлу quizzes.json у Firebase Storage та повертає їх у вигляді списку за мовами

```
export const getQuizzes = async (): Promise<{ [key in LanguageCode]: QuizTopic[] }> => {
  const file = getStorage().bucket().file('quizzes.json');
  const contents = await file.download();
  return JSON.parse(contents.toString()) as { [key in LanguageCode]: QuizTopic[] };
};
```

У даному фрагменті описано тип UserProfile, що визначає повну структуру облікового запису користувача в системі. Профіль включає основні дані, мовні налаштування, прогрес у тестах, стан рахунку та

інформацію про нагороди. Кількісні значення, такі як кількість балів або кількість монет, зберігаються окремо для кожної мови, що дозволяє підтримувати багатомовне навчання. Окрім опису типів, представлено набір утилітних функцій для отримання даних з об’єкта профілю, зокрема функції getScore, getMoney, getCompletedQuizzes та інші. Ці функції спрощують доступ до конкретних частин структури профілю відповідно до вибраної мови. Увесь опис функціональності наведено в лістингу 3.6.

Лістинг 3.6 – Програмний код, що описує структуру профілю користувача платформи, а також набір функцій для отримання даних про бали, валюту та прогрес у тестах

```
export type UserProfile = {
  id: string, auth0Id: string, name: string, email: string,
  score: { [key in LanguageCode]: number },
  money: { [key in LanguageCode]: number },
  completedQuizzes: { [key in LanguageCode]: string[] },
  openedQuizzes: { [key in LanguageCode]: string[] },
  avatar: string, language: LanguageCode,
  eligibleAwards: string[], claimedAwards: string[]};
export type LanguageCode = "en" | "es" | "fr" | "de" | "pl";
export const getScore = (user: UserProfile) =>
user.score[user.language];
export const getMoney = (user: UserProfile) =>
user.money[user.language];
export const getScoreInLanguage = (user: UserProfile,
language: LanguageCode) => user.score[language];
export const getCompletedQuizzes = (user: UserProfile) =>
user.completedQuizzes[user.language];
export const getAllCompletedQuizzes = (user: UserProfile)
=> Object.values(user.completedQuizzes).flat();
export const getOpenedQuizzes = (user: UserProfile) =>
user.openedQuizzes[user.language];
```

У цьому коді реалізовано метод getUser, який дозволяє знайти користувача в базі даних Firestore за унікальним ідентифікатором auth0Id. Виконується фільтрація колекції users з обмеженням на перший результат, після чого отриманий документ перетворюється у формат типу User. Уся логіка пошуку та обробки відповіді представлена в лістингу 3.7.

Лістинг 3.7 – Програмний код, що виконує пошук користувача за auth0Id у колекції Firestore users

```
public async getUser(auth0Id: string): Promise<User|null>
{
  const snapshot = await
firestore().collection('users').where('auth0Id', '==',
auth0Id).limit(1).get();
  if (snapshot.empty) return null;
  const doc = snapshot.docs[0];
  return { id: doc.id, ...doc.data() } as User;
}
```

Метод openQuiz реалізує логіку відкриття доступу користувача до платного тесту. Спочатку виконується завантаження всієї структури тестів з подальшим пошуком конкретного тесту за ідентифікатором quizId. Якщо тест знайдено, ініціюється транзакція Firestore, під час якої перевіряється наявність користувача, факт відкриття тесту раніше, а також достатність внутрішньої валюти на рахунку. У разі відповідності всім умовам, з рахунку списується вартість тесту, а його ідентифікатор додається до списку відкритих тестів користувача. Після завершення транзакції метод повторно завантажує оновлену інформацію про користувача та повертає її як результат. Таким чином, цей код забезпечує атомарну зміну балансу і статусу доступу до тесту, запобігаючи паралельним конфліктам. Уся реалізація логіки доступу та оновлення даних подана в лістингу 3.8.

Лістинг 3.8 – Програмний код, що відкриває доступ користувачу до тесту, списує валюту за покупку та оновлює список відкритих тестів

```
    public async openQuiz(auth0Id: string, quizId: string): Promise<User | null> {
      const quizzes = await getQuizzes();
      const quiz = Object.values(quizzes).flat().flatMap(t => t.quizzes).find(q => q.id === quizId);
      if (!quiz) return null;
      await firestore().runTransaction(async (tx) => {
        const qs = await tx.get(firestore().collection('users').where('auth0Id', '==', auth0Id).limit(1));
        if (qs.empty || quiz.price === 0) return;
        const doc = qs.docs[0];
        const user = { id: doc.id, ...doc.data() } as User;
        if (user.openedQuizzes[user.language].includes(quizId)) throw new Error('Quiz already opened');
        if (quiz.price > user.money[user.language]) throw new Error('Not enough money');
        tx.update(doc.ref, {
          money: { ...user.money, [user.language]: user.money[user.language] - quiz.price },
          openedQuizzes: { ...user.openedQuizzes, [user.language]:
user.openedQuizzes[user.language].concat(quizId) },
        });
      });
      const snapshot = await firestore().collection('users').where('auth0Id', '==', auth0Id).limit(1).get();
      if (snapshot.empty) return null;
      const doc = snapshot.docs[0];
      return { id: doc.id, ...doc.data() } as User;}
```

Метод completeQuiz реалізує повний цикл завершення тесту користувачем із подальшим оновленням його профілю. Після отримання поточних даних про тести та нагороди система знаходить тест за заданим quizId і перевіряє, чи він доступний для поточного користувача. Якщо умови дотримано, у межах транзакції оновлюються показники балів, кількість валюти та список завершених тестів. Далі відбувається аналіз можливих досягнень, які ще не були отримані, але могли стати доступними після нового прогресу.

Перевірка охоплює різні типи нагород: за кількість завершених тестів загалом, за темами, а також з урахуванням мови навчання. У разі виконання умов відповідні досягнення додаються до списку доступних, а оновлені дані зберігаються у Firestore – повна реалізація цієї логіки подана в лістингу 3.9.

Лістинг 3.9 – Програмний код, що завершує тест користувача, нараховує нагороду, додає тест до завершених та перевіряє, чи відкрито нові досягнення

```
    public async completeQuiz(auth0Id: string, quizId: string): Promise<User | null> {
      const awards = await getAwards();
      const quizzes = await getQuizzes();
      const quiz = Object.values(quizzes).flat().flatMap(t => t.quizzes).find(q => q.id === quizId);
      if (!quiz) return null;
      await firestore().runTransaction(async (tx) => {
        const qs = await tx.get(firestore().collection('users').where('auth0Id', '==', auth0Id).limit(1));
        if (qs.empty) return null;
        const doc = qs.docs[0];
        const user = { id: doc.id, ...doc.data() } as User;
        if (quiz.price !== 0 && !user.openedQuizzes[user.language].includes(quizId)) throw new Error('Quiz not opened');
```

Продовження лістингу 3.9

```
        if
(user.completedQuizzes[user.language].includes(quizId))   throw
new Error('Quiz already completed');
        const updated = {
          score:     {     ...user.score,     [user.language]:
user.score[user.language] + quiz.score },
          money:     {     ...user.money,     [user.language]:
user.money[user.language] + quiz.reward },
          completedQuizzes:     {     ...user.completedQuizzes,
[user.language]:
user.completedQuizzes[user.language].concat(quizId) },
        };
        const     notFinished     =     awards.filter(a     =>
!user.eligibleAwards.includes(a.id)                              &&
!user.claimedAwards.includes(a.id));
        const newlyFinished = notFinished.filter(a => {
          const                        all                        =
Object.values(updated.completedQuizzes).flat();
          if   (a.type   ===   'completedQuizzesAmount')   return
all.length >= a.threshold;
          if (a.type === 'completedQuizTopicsAmount') {
            const topics = Object.values(quizzes).flat();
            return  topics.filter(t  =>  t.quizzes.every(q  =>
all.includes(q.id))).length >= a.threshold;
          }
          if  (a.type  ===  'completedQuizzesAmountByLanguage')
return      updated.completedQuizzes[a.language].length      >=
a.threshold;
          if                    (a.type                    ===
'completedQuizTopicsAmountByLanguage') {
            const topics = quizzes[a.language];
            return  topics.filter(t  =>  t.quizzes.every(q  =>
all.includes(q.id))).length >= a.threshold;}
          return false;});
```

Продовження лістингу 3.9

```
        tx.update(doc.ref, newlyFinished.length > 0 ? {
          ...updated,
          eligibleAwards:             [...user.eligibleAwards,
...newlyFinished.map(a => a.id)],
        } : updated);
      });

      const            snapshot            =            await
firestore().collection('users').where('auth0Id',          '==',
auth0Id).limit(1).get();
      if (snapshot.empty) return null;
      const doc = snapshot.docs[0];
      return { id: doc.id, ...doc.data() } as User;
    }
```

Метод getLeaders виконує запит до колекції users у Firestore з метою отримання десяти користувачів з найвищим показником балів для заданої мови. Сортування здійснюється за полем score мови у спадному порядку. Якщо жодного користувача не знайдено, метод повертає порожній масив. Повна логіка формування рейтингу користувачів наведена в лістингу 3.10.

Лістинг 3.10 – Програмний код, що отримує список топ-10 користувачів за кількістю балів з певної мови

```
    public   async   getLeaders(languageCode:   LanguageCode):
Promise<User[]> {
      const            snapshot            =            await
firestore().collection('users').orderBy(`score.${languageCode}
`, 'desc').limit(10).get();
      if (snapshot.empty) return [];
      return   snapshot.docs.map(doc   =>   ({   id:   doc.id,
...doc.data() } as User));
    }
```

Метод setLanguage призначений для зміни поточної мови навчання користувача у базі даних Firestore. Після пошуку користувача за його унікальним ідентифікатором auth0Id виконується оновлення поля language. Після цього профіль повторно зчитується та повертається у вигляді об’єкта типу User. Уся процедура мовної переваги реалізована в лістингу 3.11.

Лістинг 3.11 – Програмний код, що оновлює мову навчання користувача у Firestore за його auth0Id і повертає оновлений профіль

```
    public async setLanguage(auth0Id: string, language: LanguageCode): Promise<User | null> {
      const snapshot = await firestore().collection('users').where('auth0Id', '==', auth0Id).limit(1).get();
      if (snapshot.empty) return null;
      const doc = snapshot.docs[0];
      await doc.ref.update({ language });
      const updated = await doc.ref.get();
      return { id: updated.id, ...updated.data() } as User;
    }
```

Метод claimAward реалізує механізм отримання користувачем нагороди за досягнення, перевіряючи при цьому її доступність та унікальність. На початку виконується пошук нагороди за її ідентифікатором у загальному списку, завантаженому з Firebase Storage. Якщо така нагорода не знайдена, процес припиняється. Далі ініціюється транзакція Firestore, у межах якої виконуються відповідні перевірки.

У разі виконання всіх умов, у профіль користувача додається ідентифікатор нагороди до списку отриманих, а також оновлюються бали та кількість внутрішньої валюти відповідно до значень, передбачених нагородою. Такий підхід дозволяє забезпечити цілісність і контрольованість процесу заохочення користувачів – уся логіка представлена в лістингу 3.12.

Лістинг 3.12 – Програмний код, що дозволяє користувачу отримати нагороду за досягнення: перевіряє право, оновлює рейтинг, валюту і додає нагороду до списку отриманих

```
    public async claimAward(auth0Id: string, awardId: string): Promise<User | null> {
      const award = (await getAwards()).find(a => a.id === awardId);
      if (!award) return null;
      await firestore().runTransaction(async (tx) => {
        const qs = await tx.get(firestore().collection('users').where('auth0Id', '==', auth0Id).limit(1));
        if (qs.empty) return null;
        const doc = qs.docs[0];
        const user = { id: doc.id, ...doc.data() } as User;
        if (!user.eligibleAwards.includes(awardId)) throw new Error('Award not eligible');
        if (user.claimedAwards.includes(awardId)) throw new Error('Award already claimed');
        tx.update(doc.ref, {
          claimedAwards: user.claimedAwards.concat(awardId),
          score: { ...user.score, [user.language]: user.score[user.language] + award.score },
          money: { ...user.money, [user.language]: user.money[user.language] + award.reward },
        });
      });
      const snapshot = await firestore().collection('users').where('auth0Id', '==', auth0Id).limit(1).get();
      if (snapshot.empty) return null;
      const doc = snapshot.docs[0];
      return { id: doc.id, ...doc.data() } as User;
    }
```

У лістингу 3.13 представлено конфігурацію API-шару клієнтської частини, що побудований з використанням бібліотеки Redux Toolkit Query. Він визначає базову адресу для запитів, налаштовує заголовки авторизації та формує набір тегів для кешування. Завдяки цьому забезпечується ефективна взаємодія з сервером та контроль над оновленням клієнтських даних після мутацій. Усі запити базуються на REST-архітектурі з використанням авторизації через токен доступу, збережений у стані додатку.

Оголошені запити охоплюють основні сутності платформи: профіль користувача, список тестів, лідерів, окремі тести, а також нагороди. Крім того, реалізовано мутації для завершення тесту, його відкриття, зміни мови інтерфейсу та отримання нагороди. Використання тегів дозволяє автоматично оновлювати відповідні частини інтерфейсу після змін. Таким чином, даний фрагмент формує централізований і типізований API-шар для зручної інтеграції з клієнтським інтерфейсом.

Лістинг 3.13 – Програмний код, що створює API-шар для клієнтської частини: реалізує запити до профілю, тестів, нагород, зміни мови, лідерів та дій користувача

```
export const api = createApi({
  reducerPath: "mainApi",
  tagTypes: ["UserProfile", "Quiz", "QuizTopic", "Leaders"],
  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;
    }}),
```

Продовження лістингу 3.13

```
      endpoints: (builder) => ({
        getProfile: builder.query<UserProfile, void>({ query: () => "/current-user/profile", providesTags: ["UserProfile"] }),
        getQuizzes: builder.query<{ [key in LanguageCode]: QuizTopic[] }, void>({ query: () => "/quizzes" }),
        getLeaders: builder.query<UserProfile[], void>({ query: () => "/users/leaders", providesTags: ["Leaders"] }),
        getQuiz: builder.query<Quiz, string>({ query: (id) => `/quizzes/${id}` }),
        completeQuiz: builder.mutation<void, string>({
          query: (id) => ({ url: `/current-user/quizzes/${id}/complete`, method: "POST" }),
          invalidatesTags: ["UserProfile", "Leaders"],
        }),
        openQuiz: builder.mutation<void, string>({
          query: (id) => ({ url: `/current-user/quizzes/${id}/open`, method: "POST" }),
          invalidatesTags: ["UserProfile"],
        }),
        changeLanguage: builder.mutation<void, LanguageCode>({
          query: (language) => ({ url: "/current-user/language", method: "POST", body: { language } }),
          invalidatesTags: ["UserProfile", "Leaders"],
        }),
        getAwards: builder.query<Award[], void>({ query: () => "/awards" }),
        claimAward: builder.mutation<void, string>({
          query: (id) => ({ url: `/current-user/awards/${id}/claim`, method: "POST" }),
          invalidatesTags: ["UserProfile", "Leaders"],
        }),
      }),
    });
```

3.3 Реалізація клієнтської частини платформи

Компонент QuizzesPage відповідає за відображення всіх доступних тем і тестів для поточної мови користувача з урахуванням його прогресу. Після завантаження профілю та тестів через асинхронні запити, система відображає список тем, у кожній з яких рендериться перелік тестів з інформацією про нагороду, кількість питань та статус доступності. Для кожного тесту виводиться мітка «Завершено», «Недоступно» або «Доступно», залежно від того, чи пройшов користувач тест, чи відкрив його за внутрішню валюту, чи має достатньо коштів для покупки. У разі недоступності тесту через недостатню кількість валюти кнопка «Відкрити» буде неактивною.

Якщо тест вже відкритий, але ще не завершений, користувач може перейти до нього за допомогою кнопки «Перейти». Таким чином, реалізується зручна ігрова навігація з гейміфікованими елементами, яка відображає поточний статус кожного тесту. Важливо, що доступність функціоналу напряму залежить від стану профілю користувача, що забезпечує персоналізований досвід. Повна логіка виведення тем і тестів з урахуванням статусу користувача наведена в лістингу 3.14.

Лістинг 3.14 – Програмний код, що відображає теми тестів з кнопками доступу або покупки, з урахуванням статусу користувача.

```
export const QuizzesPage: FC = () => {
  const navigate = useNavigate();
  const { data: allTopics, isLoading: l1, isError: e1 } =
useGetQuizzesQuery();
  const { data: profile, isLoading: l2, isError: e2 } =
useGetProfileQuery();
  const [openQuiz, { isLoading }] = useOpenQuizMutation();
  if (l1 || l2) return <LoadingOverlay />;
  if (e1 || e2 || !allTopics || !profile) return <PageError>
  const topics = allTopics[profile.language];
```

Продовження лістингу 3.14

```
      return (
        <PageTitle title="Тести">
          <PageBreadcrumbs items={[{ label: "Тести", href:
"/quizzes", isCurrent: true }]} />
          <PageHeader title="Тести" />
          <Box sx={{ py: 2 }}>
            {topics.map((t) => (
              <Box key={t.id} sx={{ mb: 4 }}>
                <Typography level="h4" sx={{ mb: 2, px: 2
}}>{`${t.icon} ${t.label}`}</Typography>
    <List variant="soft" sx={{ px: 2, borderRadius: "sm" }}>
                  {t.quizzes.map((q, i) => (
                    <React.Fragment key={q.id}>
                      <ListItem>
    <Box sx={{ flexGrow: 1, display: "flex", gap: 2 }}>
                        <ListItemDecorator>
    {getCompletedQuizzes(profile).includes(q.id)
                          ? <Chip size="sm"
color="success" variant="outlined">Завершено</Chip>
    : q.price > 0 && !getOpenedQuizzes(profile).includes(q.id)
                          ? <Chip size="sm" color="danger"
variant="outlined">Недоступно</Chip>
                          : <Chip size="sm"
color="primary" variant="outlined">Доступно</Chip>}
                        </ListItemDecorator>
    <Box sx={{ flexGrow: 1, opacity: q.price > 0 &&
!getOpenedQuizzes(profile).includes(q.id) ? 0.6 : 1 }}>
    <Typography level="body-sm">Награда: {q.reward} 💎
{q.score} ⭐</Typography>
    <Typography level="body-xs" sx={{ color: "text.secondary"
}}>Питань: {q.questions.length}</Typography>
                        </Box>
                      </Box>
    {q.price > 0 && !getOpenedQuizzes(profile).includes(q.id)
```

Продовження лістингу 3.14

```
? <Button size="sm" disabled={getMoney(profile) < q.price} loading={isLoading} onClick={() => openQuiz(q.id)} sx={{ ml: 2 }}>Відкрити - {q.price} 💎</Button>
: getCompletedQuizzes(profile).includes(q.id) ? null
: <Button size="sm" variant="plain" onClick={() => navigate(`/quizzes/${q.id}`)} sx={{ ml: 2 }}>Перейти</Button>}
                        </ListItem>
                        {i<t.quizzes.length - 1&&<ListDivider/>}
                      </React.Fragment>
                    ))}
                  </List>
                </Box>
              ))}
            </Box>
          </PageTitle>
        );
      };
```

Компонент QuizPage реалізує повний процес проходження тесту користувачем. Після завантаження даних про конкретний тест і профіль, перевіряється, чи було вже завершено даний тест. Якщо ні, користувачеві послідовно відображаються усі запитання з варіантами відповідей у вигляді радіо-груп. При натисканні кнопки «Завершити» система перевіряє правильність обраних відповідей, і якщо всі відповіді правильні — надсилає запит на завершення тесту, після чого перенаправляє користувача на сторінку зі списком тестів.

Якщо ж хоча б одна відповідь неправильна, відображається повідомлення про помилку під відповідним запитанням. У разі, коли тест уже завершено, інтерфейс відображає повідомлення з підтвердженням цього факту. Увесь механізм проходження тесту, перевірки та збереження результату реалізовано в лістингу 3.15.

Лістинг 3.15 – Програмний код, що реалізує проходження тесту: показує питання з варіантами відповідей, перевіряє правильність і надсилає запит на завершення

```
export const QuizPage: React.FC = () => {
  const { id } = useParams<{ id: string }>();
  const navigate = useNavigate();
  const { data: quiz, isLoading: l1, isError: e1 } = useGetQuizQuery(id!);
  const { data: profile, isLoading: l2, isError: e2 } = useGetProfileQuery();
  const [completeQuiz, { isLoading }] = useCompleteQuizMutation();
  const { control, handleSubmit, setError, formState: { errors } } = useForm<FormState>({ defaultValues: { answers: [] } });
  if (l1 || l2) return <LoadingOverlay />;
  if (e1 || e2 || !quiz || !profile) return <PageError />;
  const isCompleted = getCompletedQuizzes(profile).includes(quiz.id);
  const onSubmit = async (v: FormState) => {
    const wrong = v.answers.reduce<number[]>((acc, a, i) => quiz.questions[i].correctId !== a ? [...acc, i] : acc, []);
    if (wrong.length) {
      wrong.forEach(i => setError(`answers.${i}`, { type: "manual", message: "Incorrect answer" }));
      return;
    }
    await completeQuiz(quiz.id);
    navigate("/quizzes");
  };
  return (
    <PageTitle title={quiz.label}>
      <PageBreadcrumbs items={[{ label: "Тести", href: "/quizzes" }, { label: quiz.label, href: "/quiz", isCurrent: true }]} />
```

Продовження лістингу 3.15

```
          <PageHeader title={quiz.label} />
          {isCompleted ? (
            <PageAlert title="Quiz completed" message="You have
already completed this quiz." type="success" />
            <Box sx={{ maxWidth: 600, mx: "auto", p: 2 }}>
              {quiz.questions.map((q, i) => (
                <Card  key={i}  variant="soft"  sx={{  mb:  2,
border:   (t)   =>   `1px   solid   ${errors.answers?.[i]   ?
t.palette.danger[500] : "transparent"}` }}>
                  <CardContent>
                    <FormControl>
                      <FormLabel><Typography   level="body-md"
sx={{ mb: 1 }}>{q.label}</Typography></FormLabel>
                      <Controller
                        control={control}
                        name={`answers.${i}`}
     rules={{ required: "Please select an answer" }}
                        render={({ field }) => (
                          <RadioGroup   value={field.value   ??
null} onChange={field.onChange}>
                            {q.answers.map(a     =>     <Radio
key={a.id} value={a.id} label={a.label} />)}
                          </RadioGroup>
     {errors.answers?.[i]  &&  <Typography  color="danger"  sx={{
mt: 1 }}>{errors.answers[i].message}</Typography>}
                    </FormControl>
                  </CardContent>
                </Card>
     <Button  variant="solid"  onClick={handleSubmit(onSubmit)}
sx={{ mt: 2 }} loading={isLoading}>Завершити</Button>
            </Box>
          </PageTitle>
     };
```

Компонент AwardsPage відповідає за відображення досягнень користувача у вигляді карток із назвою нагороди, описом умов її отримання, кількістю балів і винагородою у внутрішній валюті. Для кожної нагороди визначається її тип, після чого через функцію mapDescription формується текстова інструкція, наприклад «Завершити 5 тестів французькою». Крім візуального оформлення картки, система перевіряє, чи є нагорода доступною для користувача та чи була вона вже отримана. Якщо умови виконано, користувач бачить активну кнопку «Отримати», яка при натисканні надсилає запит на зарахування нагороди.

У разі, якщо нагорода вже була отримана, кнопка переходить у стан «Отримано» та стає неактивною. Завдяки цьому інтерфейс забезпечує зрозумілу індикацію досягнень та прозорий механізм взаємодії з нагородною системою. Використання референсу ref дозволяє відстежувати, яка саме нагорода наразі обробляється, і показувати індикатор завантаження тільки для неї. Повна логіка відображення нагород, перевірки умов та обробки взаємодії з користувачем представлена в лістингу 3.16.

Лістинг 3.16 – Програмний код, що реалізує сторінку досягнень: показує нагороди, їх опис, стан доступності та кнопку для отримання, якщо умови виконано

```
export const AwardsPage: FC = () => {
  const { data: awards, isLoading: l1, isError: e1 } =
useGetAwardsQuery();
  const { data: profile, isLoading: l2, isError: e2 } =
useGetProfileQuery();
  const [claimAward, { isLoading }] =
useClaimAwardMutation();
  const ref = useRef<string | null>(null);
  if (l1 || l2) return <LoadingOverlay />;
  if (e1 || e2 || !awards || !profile) return <PageError/>;
  return (
    <PageTitle title="Досягнення">
```

Продовження лістингу 3.16

```
          <PageBreadcrumbs items={[{ label: "Досягнення", href:
"/awards", isCurrent: true }]} />
          <PageHeader title="Досягнення" />
          <Box  p={2}  display="flex"  flexWrap="wrap"  gap={2}
justifyContent="center">
            {awards.map((a) => (
              <Card key={a.id} variant="outlined" sx={{ width:
320, flex: "1 1 320px", pb: 0 }}>
                <CardContent>
    <Typography level="body-lg">{a.title}</Typography>
                  <Typography  level="body-sm"  sx={{  mt:  1
}}>{mapDescription(a)}</Typography>
                </CardContent>
                <Stack>
                  <CardOverflow variant="soft" sx={{ bgcolor:
"background.level1" }}>
                    <Divider inset="context" />
                    <CardContent orientation="horizontal">
                      <Box     display="flex"     width="100%"
justifyContent="center" gap={4}>
                        <Typography            level="body-md"
textColor="text.secondary">{a.score} ⭐</Typography>
                        <Typography            level="body-md"
textColor="text.secondary">{a.reward} 💎</Typography>
                    </CardContent>
                  </CardOverflow>
                  {profile.eligibleAwards.includes(a.id) && (
                    <CardOverflow     variant="soft"     sx={{
bgcolor: "background.level1" }}>
                      <Divider inset="context" />
                      <CardContent orientation="horizontal">
    {profile.claimedAwards.includes(a.id) ? (
                        <Button  variant="solid"  fullWidth
disabled>Отримано</Button>
```

Продовження лістингу 3.16

```
                        <Button variant="solid" fullWidth
onClick={() => { ref.current = a.id; claimAward(a.id); }}
loading={isLoading && ref.current === a.id}>
                          Отримати
                        </Button>
                      )}
                    </CardContent>
                  </CardOverflow>
                )}
              </Stack>
            </Card>
          ))}
        </Box>
      </PageTitle>
    );
  };

  const mapLanguageName = (code: LanguageCode) => {
    switch (code) {
      case "es": return "іспанською";
      case "fr": return "французькою";
      case "de": return "німецькою";
      case "pl": return "польською";
      default: return "англійською";
    }
  };
```

Компонент LeadersPage реалізує сторінку з відображенням топ-10 користувачів за кількістю набраних балів відповідно до обраної мови навчання. Для цього спочатку завантажуються дані профілю поточного користувача, а також список лідерів, відсортованих за спаданням балів. У разі успішного отримання обох відповідей, ініціалізується виведення

таблиці лідерів за допомогою компонента LeadersList, якому передається масив користувачів та мова інтерфейсу.

Якщо дані ще завантажуються, на екрані показується індикатор LoadingOverlay, а у разі помилки – повідомлення PageError. Компонент відображає заголовок сторінки та хлібні крихти для навігації, що підтримує загальний стиль платформи. Таким чином, сторінка є точкою доступу до рейтингу користувачів, формуючи конкурентне середовище на основі прогресу. Повна реалізація логіки сторінки рейтингу представлена в лістингу 3.17.

Лістинг 3.17 – Програмний код, що реалізує сторінку з таблицею лідерів: отримує профіль користувача та список лідерів, і виводить їх через компонент LeadersList

```
export const LeadersPage: FC = () => {
  const { data: leaders, isLoading: l1, isError: e1 } = useGetLeadersQuery();
  const { data: profile, isLoading: l2, isError: e2 } = useGetProfileQuery();
  if (l1 || l2) return <LoadingOverlay />;
  if (e1 || !leaders || e2 || !profile) return <PageError/>;
  return (
    <PageTitle title="Таблиця лідерів">
      <PageBreadcrumbs items={[{ label: "Таблиця лідерів", href: "/leaders", isCurrent: true }]} />
      <PageHeader title="Таблиця лідерів" />
      <Box sx={{ p: 2, minWidth: 240 }}>
        <LeadersList leaders={leaders} language={profile.language} />
      </Box>
    </PageTitle>
  );
};
```

Компонент LeadersList виводить список лідерів у вигляді вертикального нумерованого переліку, де кожен учасник представлений з аватаром, ім’ям та кількістю балів для обраної мови. Усі записи відображаються в стилі soft-списку з роздільниками між елементами. Поточний користувач позначається міткою «ВИ», що дозволяє швидко зорієнтуватися у власному місці в рейтингу. Увесь механізм відображення рейтингу реалізовано в лістингу 3.18.

Лістинг 3.18 – Програмний код, що відображає список лідерів у вигляді нумерованого переліку з аватаром, ім’ям, балами та міткою для поточного користувача

```
export const LeadersList: FC<LeadersListProps> = ({ leaders, language }) => {
  const { user } = useAuth0();
  if (!user) return null;
  return (
    <List variant="soft" sx={{ borderRadius: "sm" }}>
      {leaders.map((l, i) => (
        <Fragment key={i}>
          <ListItem>
            <ListItemDecorator><Avatar size="sm" src={l.avatar} /></ListItemDecorator>
<Box sx={{flexGrow: 1,display: flex,gap: 1, alignItems}}>
<Typography level="body-sm">{i + 1}. {l.name}</Typography>
              {l.email === user.email && (
                <Chip size="sm" color="primary" variant="soft" sx={{ ml: 1 }}>ВИ</Chip>)}
            <Typography level="body-md">{getScoreInLanguage(l, language)} ⭐</Typography>
          </ListItem>
          {i < leaders.length - 1 && <ListDivider />}
        </Fragment> };
```

Компонент ProfilePage реалізує функціональну сторінку профілю користувача з підтримкою трьох вкладок: персональні дані, прогрес у навчанні та вибір мови. Після завантаження профілю і структури тестів здійснюється обчислення завершених тестів користувача, які використовуються у вкладці прогресу. Вкладка «Персональна інформація» відображає аватар, ім’я та електронну пошту користувача у формі, яка недоступна для редагування. Завдяки інтеграції з Auth0 забезпечується безпечна автентифікація користувачів і доступ до відповідних даних.

На вкладці «Прогрес» відображаються завершені тести разом з кількістю набраних балів та іконкою мови, якою проходився кожен тест. Якщо тести відсутні – користувачу показується відповідне повідомлення. Третя вкладка «Мова» дозволяє змінити мову інтерфейсу і навчального контенту через список радіокнопок з прапорами відповідних країн. Повна логіка взаємодії з профілем, прогресом та мовними налаштуваннями реалізована в лістингу 3.19.

Лістинг 3.19 – Програмний код, що реалізує сторінку профілю з трьома вкладками: перегляд персональних даних, список завершених тестів із балами й мовами, а також зміна поточної мови навчання

```
export const ProfilePage: FC = () => {
  const { data: profile, isLoading: l1, isError: e1 } = useGetProfileQuery();
  const { data: quizzes, isLoading: l2, isError: e2 } = useGetQuizzesQuery();
  const [changeLanguage] = useChangeLanguageMutation();
  const [params, setParams] = useSearchParams();
  const tab = params.get("tab") ?? "personal";
  const { user } = useAuth0();
  if (!user || l1 || l2) return <LoadingOverlay />;
  if (e1 || e2 || !profile || !quizzes) return <PageError>;
  const avatar = profile.avatar ?? `https://i.pravatar.cc/40?u=${user.email}`;
```

Продовження лістингу 3.19

```
      const allQuizzes = Object.entries(quizzes).flatMap(([lang, topics]) =>
        topics.flatMap(t => t.quizzes.map(q => ({ ...q, language: lang }))));
      const completed = getAllCompletedQuizzes(profile).map(id => allQuizzes.find(q => q.id === id)!);
      return (
        <PageTitle title="Профіль">
          <PageBreadcrumbs items={[{ label: "Профіль", href: "/profile", isCurrent: true }]} />
          <PageHeader title="Профіль" />
          <Tabs value={tab} onChange={(_, v) => setParams({ tab: String(v) }, { replace: true })}>
            <TabList>{TABS.map(t => <Tab key={t.value} value={t.value}>{t.label}</Tab>)}</TabList>
          </Tabs>
          {tab === "personal" && (
            <Card variant="soft">
              <Typography>Персональна інформація</Typography>
              <Stack direction="row" spacing={3}>
                <AspectRatio ratio="1"><img src={avatar} loading="lazy" alt="" /></AspectRatio>
                <Stack spacing={2} flexGrow={1}>
                  <Input value={profile.name} readOnly />
                  <Input value={profile.email} readOnly type="email" startDecorator={<EmailRoundedIcon />} />
                </Stack>
              </Stack>
            </Card>
          )}
          {tab === "progress" && (
            completed.length > 0 ? (
              <Box display="flex" flexWrap="wrap" gap={2}>{completed.map(q =>
```

Продовження лістингу 3.19

```
                <Card key={q.id}><CardContent><Link
href={`/quizzes/${q.id}`}>{q.label}</Link></CardContent>
                <CardOverflow><CardContent>{q.score}
<CountryFlag
countryCode={mapLanguageCodeToCountryCode(q.language)}
/></CardContent></CardOverflow></Card>
              )}</Box>
            )      :      <Typography>Немає      завершенних
тестів</Typography>
          )}
          {tab === "language" && (
            <RadioGroup value={profile.language} onChange={e
=> changeLanguage(e.target.value as LanguageCode)}>
              <List>{languages.map(l =>
                <ListItem
key={l.value}><ListItemDecorator>{l.icon}</ListItemDecorator>
                <Radio overlay value={l.value} label={l.label}
variant="outlined" /></ListItem>
              )}</List>
            </RadioGroup>
          )}
        </PageTitle>
      );
    };
```

Реалізація гейміфікованої платформи для вивчення мов була успішно завершена шляхом створення повнофункціонального вебзастосунку з чіткою архітектурою та сучасним інтерфейсом. Серверна частина, розроблена з використанням Nest.js, забезпечує надійне управління користувачами, тестами, прогресом і нагородами. Клієнтська частина, створена на базі React з використанням бібліотеки Joy UI, гарантує зручний користувацький досвід та візуальну привабливість інтерфейсу.

У системі реалізовано механізм проходження тестів з автоматичною перевіркою відповідей і нарахуванням винагороди. Користувачі мають змогу переглядати свій прогрес, змінювати мову навчання, відкривати платні тести, отримувати досягнення та конкурувати з іншими в таблиці лідерів. Для зберігання даних використано хмарну базу Firestore та Firebase Storage, що забезпечує масштабованість і збереження актуального стану системи.

Особливу увагу було приділено аспектам гейміфікації, що сприяє залученню користувачів до регулярного навчання. Інтеграція нагород, зірок, алмазів і рейтингу мотивує продовжувати взаємодію з платформою. Завдяки застосованому стеку технологій реалізовано гнучку, розширювану і продуктивну систему, що може бути основою для подальшого масштабування або додавання нових функцій.

3.4 Побудова гейміфікаційних механізмів

Гейміфікація є ключовим елементом платформи, що безпосередньо впливає на мотивацію користувачів і сприяє підвищенню залучення до процесу навчання. У розробленій системі гейміфікаційні механізми реалізовані на декількох рівнях взаємодії. Насамперед, кожен тест має визначену кількість балів та винагороду у вигляді внутрішньої валюти – алмазів. Бали відображають рівень успішності користувача у таблиці лідерів, тоді як алмази дозволяють відкривати нові тести, тим самим створюючи економічну модель прогресу.

Користувач отримує бали і винагороду лише після успішного проходження тесту, коли всі відповіді є правильними. У випадку, якщо тест є платним, спочатку необхідно витратити певну кількість алмазів для його відкриття. Всі ці дії супроводжуються оновленням стану профілю в реальному часі. Таким чином, користувач постійно відчуває вплив своїх

рішень і має стимул вдосконалюватися. Наявність умов доступу і винагороди формує відчуття значущості кожного досягнення.

Окрему роль у платформі відіграють досягнення. Це заздалегідь визначені цілі, які користувач може досягти, виконуючи певні дії, наприклад, завершивши вказану кількість тестів або тем, загалом або для конкретної мови. Система автоматично перевіряє, чи були виконані умови для отримання нової нагороди. Якщо так – вона додається до списку доступних для отримання, і користувач може її активувати, отримавши додаткові бали та алмази. Таким чином, формується відчуття поступу, а платформа перетворюється на навчальну гру з елементами змагання.

Додатковим гейміфікаційним елементом є таблиця лідерів, де відображається топ-10 користувачів за кількістю набраних балів у межах певної мови. Це створює дух суперництва і заохочує користувачів повертатися до платформи з метою покращення свого результату. Поточний користувач також позначений у списку, що дозволяє йому швидко оцінити власне місце серед інших. Усі ці механізми працюють узгоджено та доповнюють одне одного, формуючи комплексну систему заохочення у процесі навчання.

3.5 Візуалізація інтерфейсу платформи

Інтерфейс платформи був реалізований з урахуванням сучасних вимог до зручності, інтуїтивності та візуальної привабливості. Основна мета полягала у створенні зрозумілої структури, яка дозволяє користувачу швидко орієнтуватися у функціоналі застосунку. Інтерфейс побудовано на основі компонентийної системи, що використовує бібліотеку компонентів joy ui, а навігація реалізована за допомогою react-router. У верхній панелі розміщені основні навігаційні пункти – тести, досягнення, таблиця лідерів – а також індикатори поточного балансу користувача у вигляді балів та алмазів.

На сторінці тестів реалізовано відображення тем та списку відповідних тестів, кожен з яких має власний статус, винагороду, ціну та кількість питань. Для кожного тесту система визначає, чи він завершений, доступний або недоступний, і відображає відповідний маркер – зелений для завершених, синій для доступних і червоний для тих, що потребують відкриття за алмази. Інтерфейс також динамічно реагує на стан користувача: наприклад, якщо не вистачає алмазів для відкриття тесту, кнопка буде неактивною. Далі можна побачити приклад відображення інтерфейсу сторінки тестів з усіма описаними елементами (рисунок 3.1).

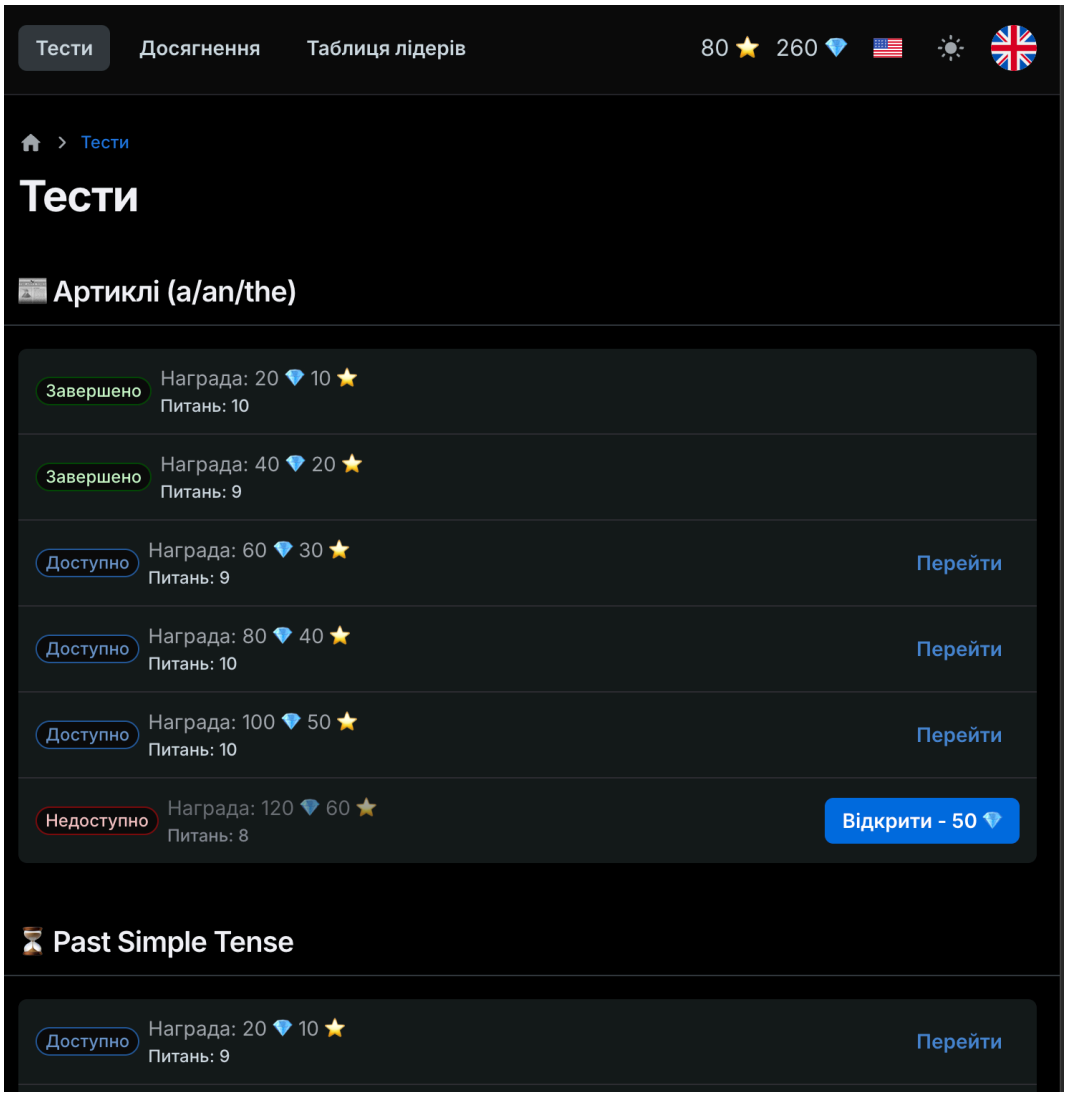

Рисунок 3.1 – Головна сторінка платформи з тестами

Далі можна побачити верхню панель інтерфейсу, яка поєднує елементи глобальної навігації, індикатори прогресу та інструменти налаштування користувацького середовища (рисунок 3.2). У лівій частині панелі розташовано посилання на основні сторінки платформи: тести, досягнення та таблицю лідерів. У правій частині відображаються кількість балів та алмазів, обрана мова інтерфейсу, кнопка перемикання теми та доступ до меню профілю користувача.

Рисунок 3.2 – Верхня панель із навігацією, балами, валютою, кнопками перемикання мови, перемикання теми інтерфейсу та меню користувача

Інтерфейс платформи підтримує зміну теми оформлення, що є важливою частиною персоналізації досвіду користувача. Далі можна побачити приклад інтерфейсу у світлій темі, яка забезпечує високу контрастність та зручність для користувачів у добре освітлених середовищах (рисунок 3.3). Візуальні акценти збережені аналогічно темній темі – кольорові індикатори доступності тестів, кнопки навігації та інформація про винагороду.

Зміна теми не впливає на розташування або логіку елементів інтерфейсу. Усі функціональні компоненти, такі як перелік тестів, їхній статус, кнопки «перейти» або «відкрити», залишаються у звичних місцях. Це дозволяє користувачам легко адаптуватися до будь-якого оформлення та не втрачати ефективність взаємодії з платформою.

Індикація статусу тестів реалізована за допомогою кольорових позначок: зелений означає завершений, синій – доступний, сірий із червоною рамкою – недоступний.

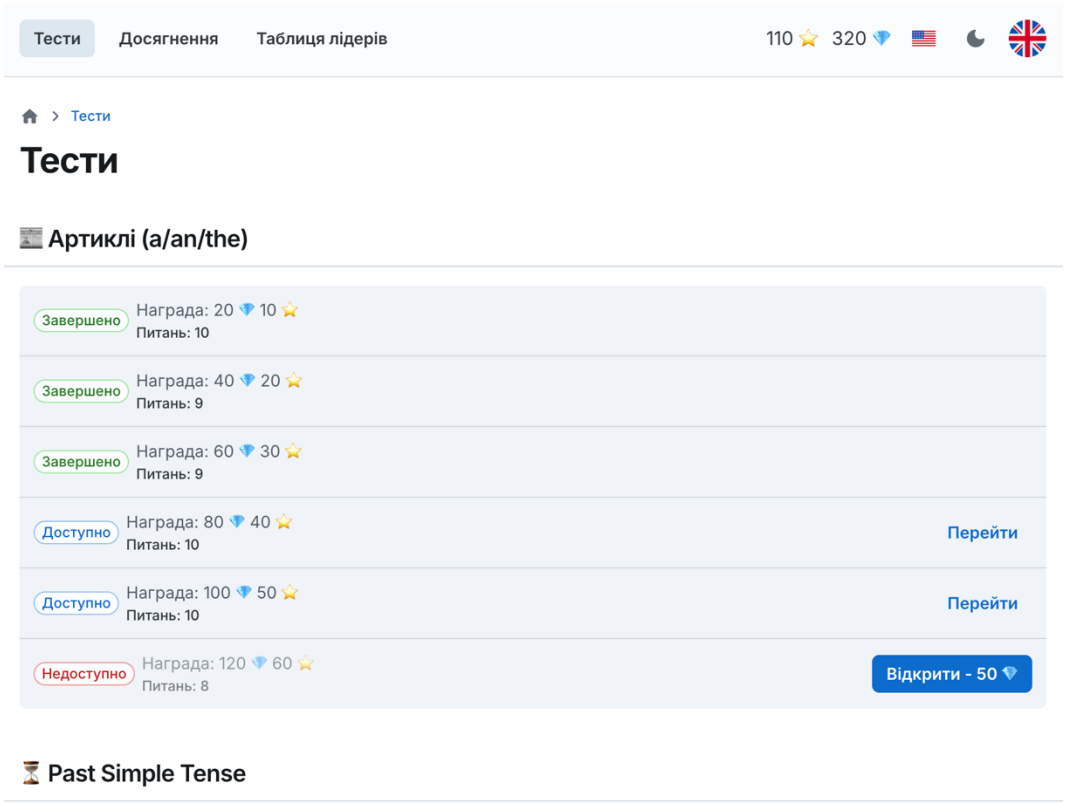

Рисунок 3.3 – Платформа у світлій темі

Інтерфейс проходження тесту має просту та інтуїтивно зрозумілу структуру, яка зосереджує увагу користувача на завданнях. Кожне запитання представлено окремою карткою з чітким формулюванням та варіантами відповіді у вигляді радіо-кнопок. Це дозволяє зручно взаємодіяти з системою та фокусуватися на контенті, мінімізуючи відволікальні елементи. Далі можна побачити реалізацію інтерфейсу проходження тесту у темному оформленні (рисунок 3.4).

Запитання згруповані вертикально, що забезпечує природний рух погляду зверху вниз. Обраний варіант відповіді чітко виділяється, а кольорове маркування після перевірки дозволяє користувачу миттєво ідентифікувати помилкові відповіді. Завдяки контрастному фону текст залишається добре читабельним навіть при тривалому використанні застосунку.

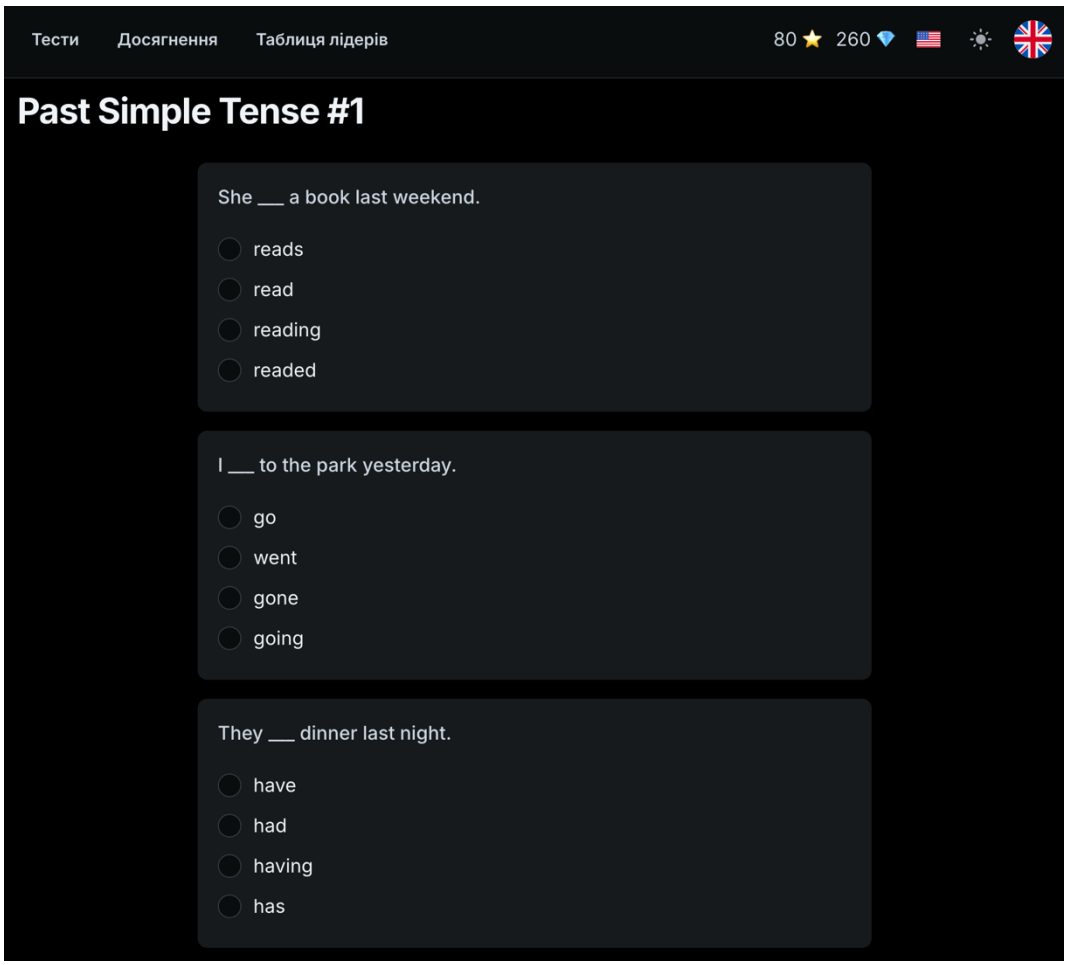

Рисунок 3.4 – Інтерфейс проходження тесту

Окрема увага в інтерфейсі проходження тесту приділена зворотному зв’язку, що допомагає користувачу навчатися на власних помилках. У разі вибору неправильної відповіді система візуально сигналізує про це червоною рамкою навколо блоку питання та текстом помилки. Такий підхід робить процес навчання більш ефективним і дає змогу одразу розпізнати помилкові дії. Далі можна побачити приклад підсвічування неправильної відповіді у тесті (рисунок 3.5).

Кожне питання представлено у компактному прямокутному блоці з темним фоном, де варіанти відповіді чітко відокремлені один від одного. Після вибору користувачем некоректного варіанту система додає підпис англійською мовою incorrect answer, що відповідає міжнародним

стандартам UX. Колірне кодування виконує роль візуального підсилення текстового повідомлення, а отже підвищує загальну зрозумілість.

Інтерфейс дозволяє залишатися в контексті тесту без потреби у перезавантаженні або переході на іншу сторінку. Це сприяє безперервному навчальному процесу та забезпечує позитивний досвід використання платформи. Така реалізація відповідає сучасним вимогам до інтерактивних освітніх застосунків і має високу ергономіку.

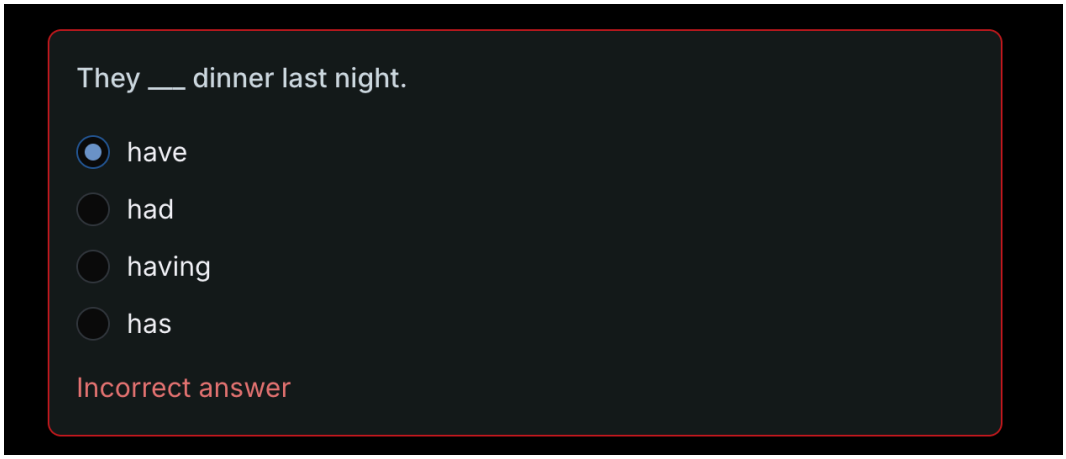

Рисунок 3.5 – Інтерфейс тесту з підсвічуванням неправильної відповіді

Інтерфейс платформи передбачає багатомовну підтримку, яка дозволяє користувачу змінювати мову навчання та отримувати доступ до тестів відповідною мовою. Це особливо актуально в умовах гейміфікованого середовища, де навчальний контент має бути адаптований до лінгвістичного контексту користувача. Далі можна побачити приклад того, як відображаються тести і теми після вибору іспанської мови як основної (рисунок 3.6).

Верхня панель динамічно відображає відповідний прапорець, а вміст сторінки оновлюється відповідно до вибраної мови. Назви тем та тестів адаптовані і доповнені іспанськими прикладами, як-от «el/la/los/las» або «Pretérito Indefinido», що допомагає користувачу зануритись у мовне

середовище. Візуальне структурування списку залишається незмінним, збережено позначення доступності, нагороди та кількості питань.

Інтерфейс також динамічно враховує прогрес у тестах для кожної мови окремо. Це дозволяє користувачеві накопичувати бали та алмази в межах обраного мовного контексту, стимулюючи проходження більшої кількості тестів на різних мовах. Такий підхід підвищує гнучкість платформи та робить її універсальним інструментом для вивчення лексики.

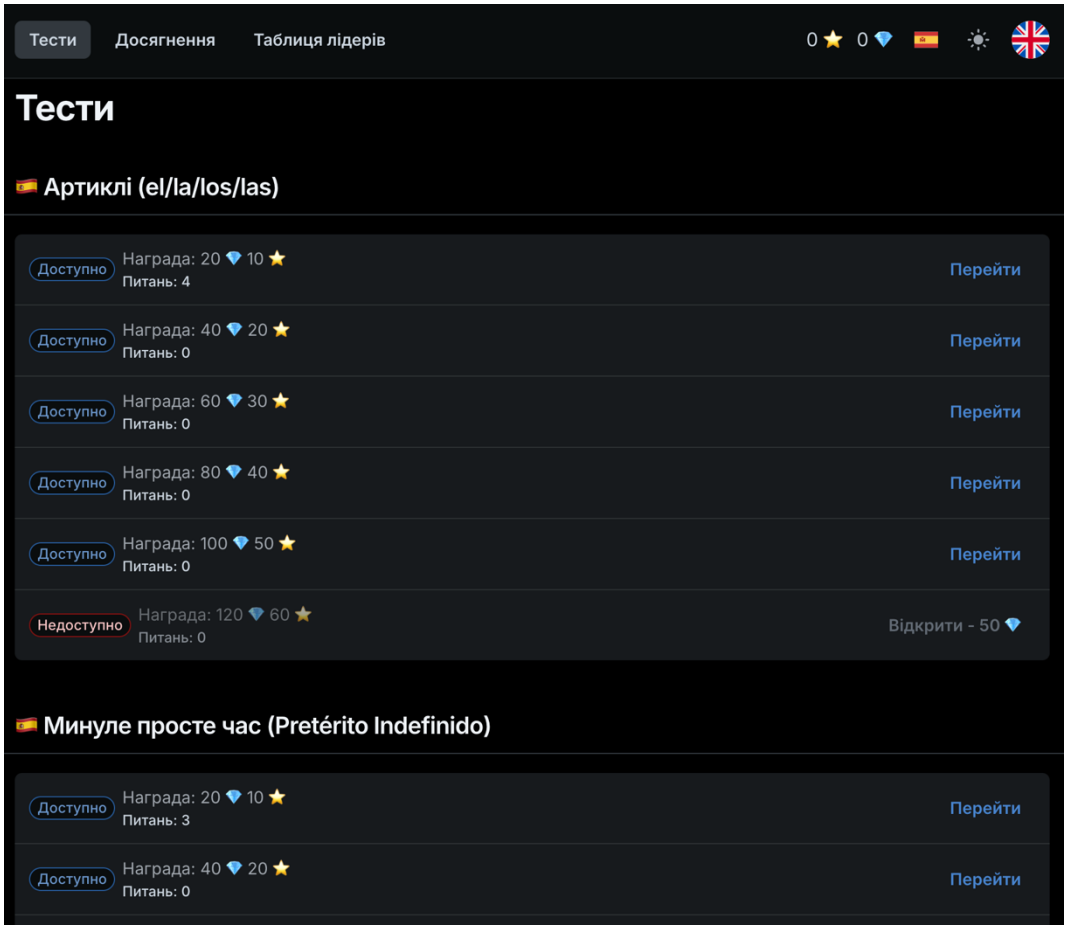

Рисунок 3.6 – Тести на іншій мові

На платформі передбачено систему досягнень, яка слугує важливим елементом гейміфікації та стимулює користувача до активної участі у навчальному процесі. Кожне досягнення має конкретну мету, як-от завершити певну кількість тестів або опанувати теми в межах вибраної

мови. Наприклад, досягнення Quiz Master передбачає проходження 50 тестів і надає винагороду у вигляді 100 балів та 500 алмазів. Далі можна побачити перелік доступних досягнень та їх статус (рисунок 3.7).

Важливою особливістю є персоналізація досягнень за мовою. Такі нагороди, як French Linguist або Spanish Scholar, вимагають завершення тестів саме французькою або іспанською мовою відповідно. Кожне досягнення містить опис умови, а також чітко позначені значення винагороди у зірках та алмазах.

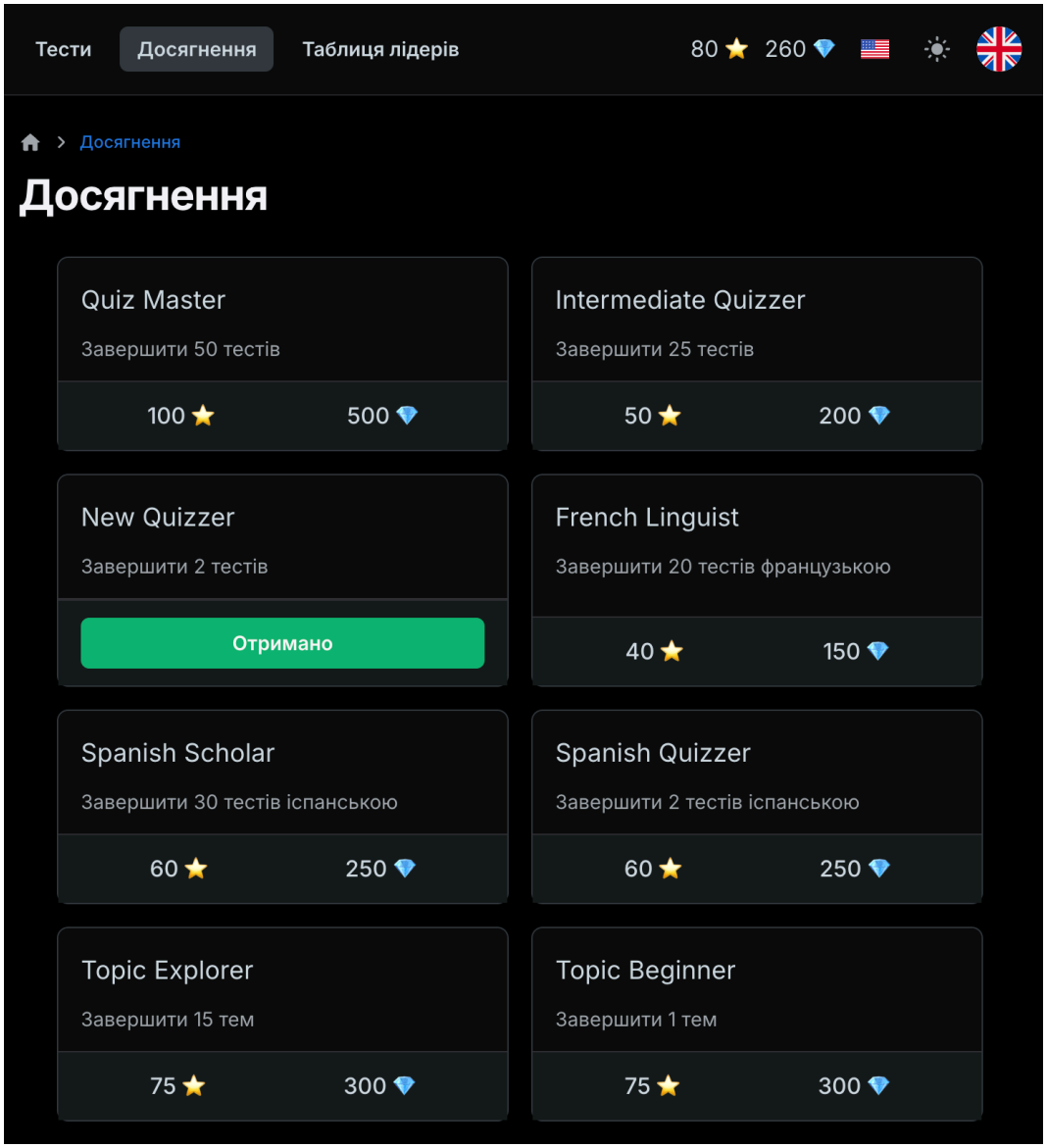

Рисунок 3.7 – Сторінка «Досягнення» та її контент

Сторінка таблиці лідерів є важливим компонентом гейміфікаційної складової платформи, що дозволяє користувачам порівнювати власні результати з досягненнями інших. Рейтинг формується на основі загальної кількості балів, отриманих у процесі проходження тестів. На першому місці перебуває користувач з найбільшою кількістю балів, що сприяє формуванню духу змагання. Далі можна побачити приклад реалізованої таблиці лідерів на платформі (рисунок 3.8).

Кожен запис у таблиці містить порядковий номер, ім’я користувача, його аватар і кількість накопичених зірок. Особливістю є те, що поточний користувач відображається з міткою «ви», що дозволяє швидко зорієнтуватися у своєму місці в рейтингу. Подібна персоналізація підвищує залученість і стимулює прагнення підвищувати результати.

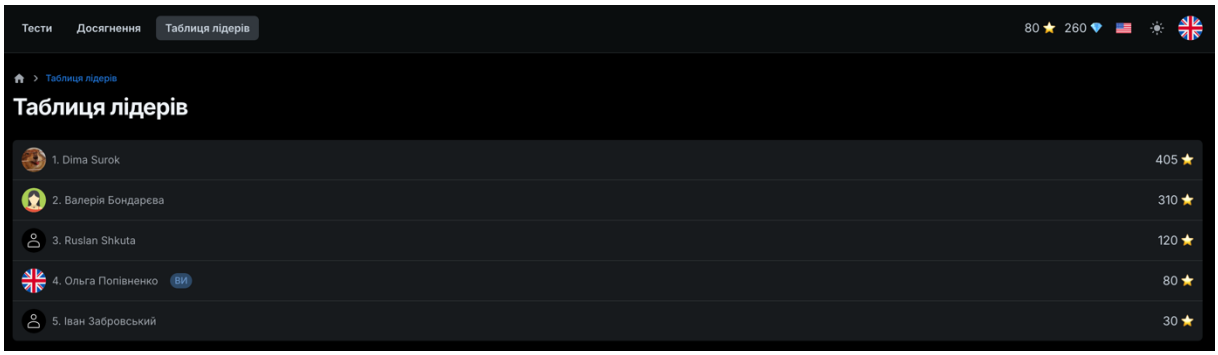

Рисунок 3.8 – Сторінка «Таблиці лідерів» та її контент

Елемент інтерфейсу у вигляді аватарки виконує роль основної точки доступу до персонального кабінету користувача. Він розміщений у верхньому правому куті навігаційної панелі та слугує як візуальний маркер поточного профілю, зображуючи прапор вибраної мови. При натисканні на аватар відкривається спливаюче меню, яке містить ім’я користувача, адресу електронної пошти та кнопку для виходу з облікового запису. Далі можна побачити вигляд цієї інтеракції (рисунок 3.9).

Рішення реалізоване таким чином, щоб користувач швидко отримував доступ до найважливішої інформації про себе. Завдяки компактності меню воно не перевантажує інтерфейс, водночас забезпечуючи всю необхідну функціональність. Важливою деталлю є використання прапора замість фото, що одночасно вказує на активну мову навчання.

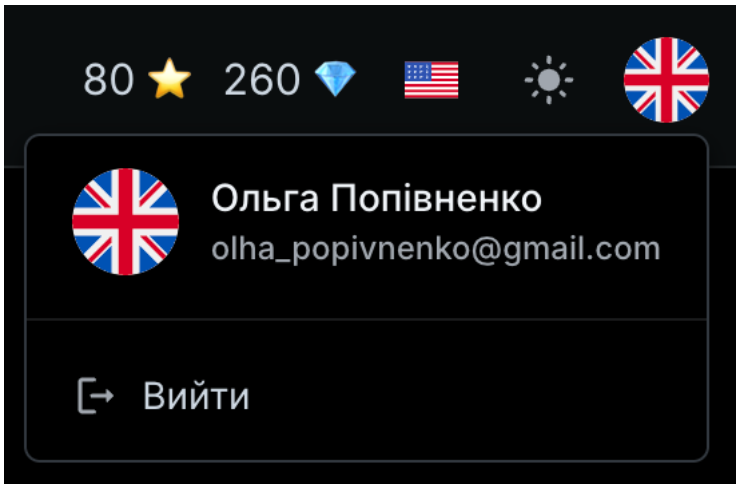

Рисунок 3.9 – Аватар-кнопка доступу до профілю

У структурі профілю користувача передбачено розділ «Персональна інформація», який відображає базові дані облікового запису, зокрема ім’я та адресу електронної пошти. Ці поля представлені у вигляді заблокованих текстових полів, що унеможливлює їх зміну напряму з інтерфейсу. Це забезпечує захист критичних даних користувача від випадкової модифікації. Далі можна побачити приклад реалізації цього розділу (рисунок 3.10).

Зліва від форми розміщено аватар з прапором, який відображає поточну вибрану мову користувача, що узгоджується з іконкою у верхній панелі. Така візуальна репрезентація дозволяє швидко зорієнтуватися у мовному контексті платформи. Оформлення всіх елементів витримано в єдиному стилі, що підсилює враження злагодженого та продуманого інтерфейсу.

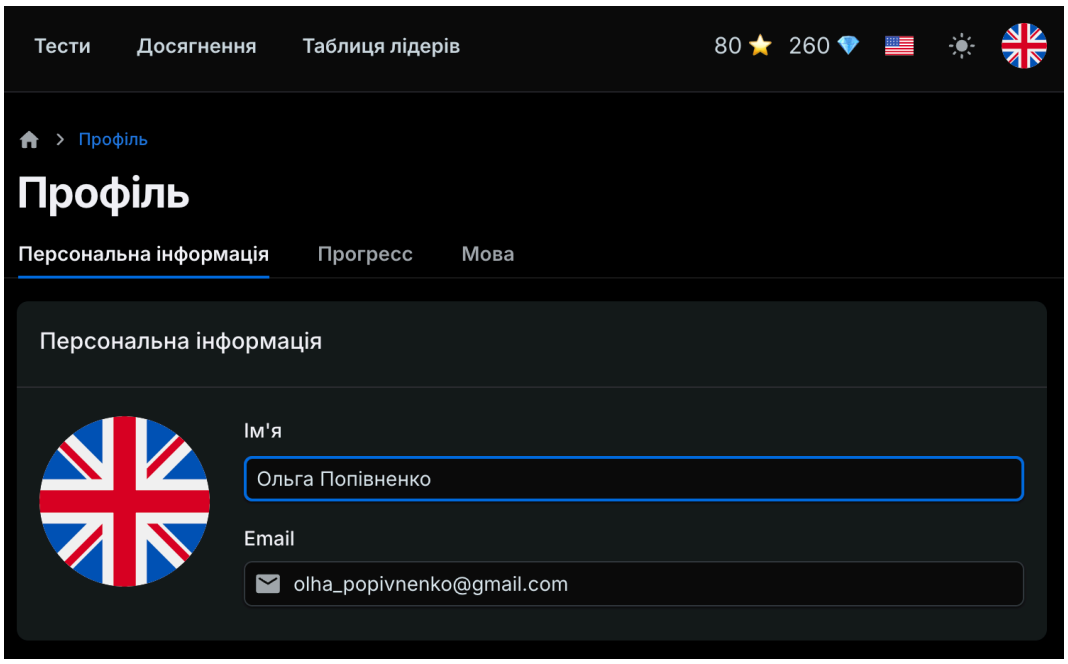

Рисунок 3.10 – Персональна інформація користувача

У вкладці «Прогрес» користувач може переглянути список тестів, які він уже завершив. Кожен елемент цього списку представлений у вигляді окремої картки, яка містить назву тесту, кількість отриманих балів та іконку мови, на якій тест був пройдений. Така структура дозволяє швидко оцінити прогрес у вивченні певної мови. Далі можна побачити приклад реалізації інтерфейсу прогресу користувача (рисунок 3.11).

Картки відрізняються за вмістом залежно від пройдених тестів, що дозволяє користувачу легко відстежувати свої досягнення у навчанні. Зображення прапора біля кожного тесту вказує на мову, яку опановував користувач під час проходження, а зірочки сигналізують про кількість набраних балів. Це мотивує повертатися до платформи для накопичення результатів і відкриття нових досягнень.

Навігація між вкладками реалізована у вигляді горизонтального меню, що дозволяє інтуїтивно перемикатися між розділами персональної інформації, прогресу та мовних налаштувань. Такий підхід дозволяє забезпечити послідовну та зручну взаємодію з платформою.

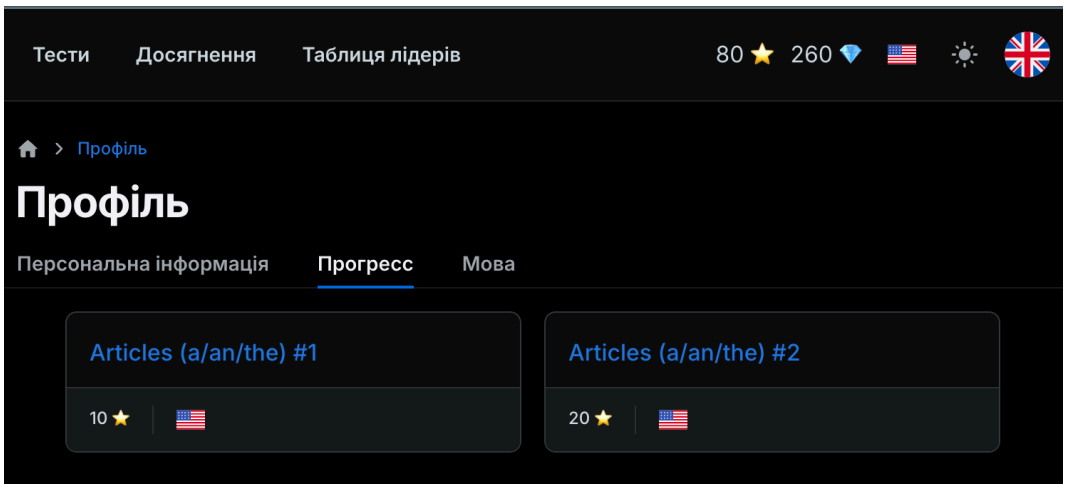

Рисунок 3.11 – Прогрес користувача

Текст У межах платформи реалізована можливість вибору мови навчання, що дозволяє користувачу адаптувати контент відповідно до власних потреб. Перемикання мови змінює доступний набір тестів, накопичення балів та віртуальної валюти, а також окремо відслідковує прогрес для кожної з обраних мов. Далі можна побачити інтерфейс вибору мови користувачем (рисунок 3.12).

Список мов відображається у вигляді вертикального радіо-групованого списку, кожен елемент якого супроводжується відповідним прапором країни. Такий підхід сприяє швидкому візуальному сприйняттю та спрощує вибір навіть для тих користувачів, що мають початковий рівень володіння мовою. У поточному прикладі активною є англійська мова, що виділяється відповідною обводкою.

Зміна мови активується при натисканні на відповідний елемент, після чого система автоматично оновлює мовні налаштування у профілі користувача. Це забезпечується інтеграцією з бекендом, який обробляє оновлення та синхронізує зміни з базою даних користувачів. Зміни застосовуються без потреби перезавантаження сторінки, що сприяє кращому досвіду взаємодії з системою.

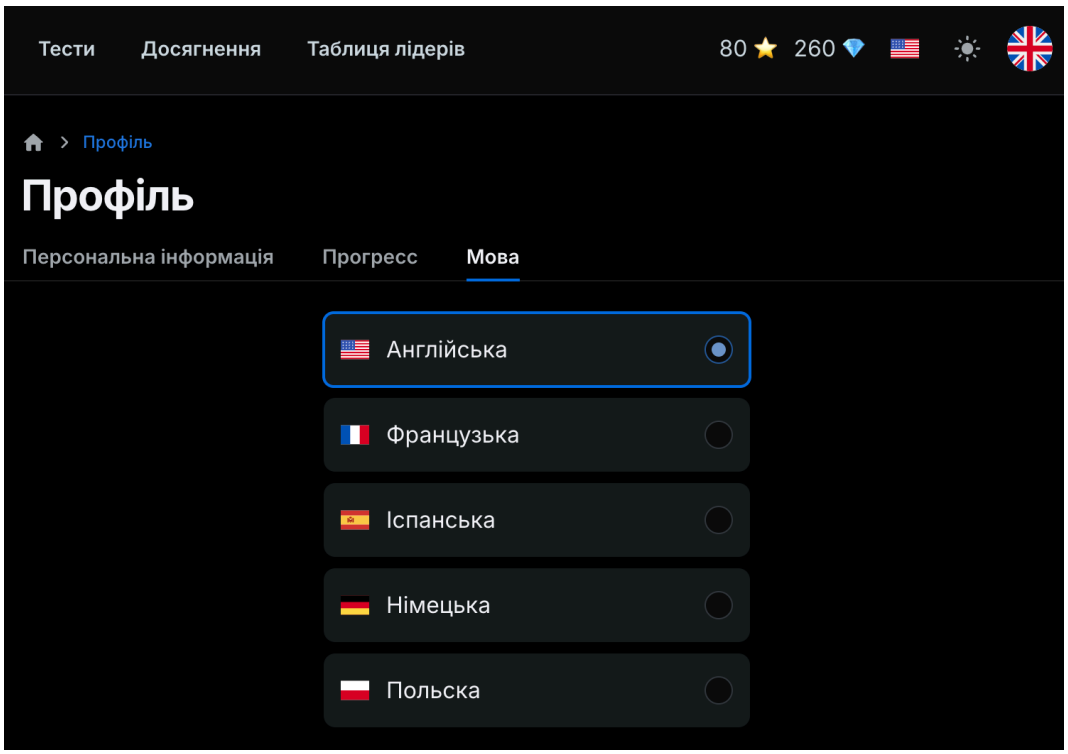

Рисунок 3.12 – Вибір мови

Завдяки можливості перемикання мов користувачі можуть проходити окремі курси, тести або теми, адаптовані під конкретну мовну групу, залишаючись при цьому в межах одного облікового запису. Це означає, що для кожної мови навчання користувач отримує окрему шкалу прогресу, персоналізований список завершених завдань, індивідуальну кількість набраних балів та внутрішньої валюти. Наприклад, якщо користувач вивчає одночасно англійську та іспанську мови, то система зберігає і розділяє його активність у кожній мовній категорії окремо, не змішуючи їх між собою, що забезпечує чітке розуміння навчальних досягнень.

Завдяки цьому функціоналу реалізується повноцінна підтримка багатомовного навчання, яка дозволяє користувачу самостійно обирати порядок опанування мов і поєднувати їх у зручному для себе темпі. Такий підхід особливо актуальний у контексті глобалізованого освітнього середовища, де користувачі можуть пересікатися з кількома мовними спільнотами або мати потребу в адаптації до нових мовних реалій.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було повністю реалізовано поставлене завдання – створення вебдодатку для проходження та гейміфікованого опрацювання тестових завдань з іноземних мов з урахуванням багатомовної підтримки та персоналізованого прогресу. Система охоплює усі ключові компоненти навчальної платформи: управління профілем користувача, багаторівневу структуру тестів, механізм відкриття та завершення тестів, нарахування балів та внутрішньої валюти, облік досягнень, таблицю лідерів, а також адаптивний інтерфейс для перемикання мови навчання. Технічно проект реалізовано на сучасному стеку технологій – клієнтська частина створена з використанням React, Joy UI, React Router, Auth0 та TypeScript, тоді як серверна частина базується на Nest.js з інтеграцією Firestore як хмарного сховища даних, а також Firebase Storage для розміщення контенту.

Досягнуті кількісні показники демонструють достатню завершеність і функціональну повноту реалізованої системи. У вебдодатку успішно реалізовано багатомовну підтримку для п’яти мов: англійської, французької, іспанської, німецької та польської, з можливістю перемикання без перезавантаження сесії. Платформа забезпечує індивідуальне збереження балів, валюти та прогресу для кожної мови, що дозволяє користувачу одночасно опрацьовувати кілька напрямків навчання. У межах демонстраційної версії було реалізовано понад 20 тестів, структурованих за темами, із різною кількістю запитань, рівнем складності та винагородою за завершення. Кожне завершене завдання автоматично перевіряється системою, а результати фіксуються у профілі користувача. На основі цього платформа автоматично розраховує, чи досягнуто умови для відкриття нових досягнень, і пропонує відповідні нагороди.

У порівнянні з аналогічними освітніми платформами, такими як Duolingo, Memrise чи Busuu, розроблена система вирізняється простотою

інтерфейсу, можливістю розширення контенту без змін у структурі бази даних, а також відкритістю архітектури для подальшого масштабування. Основний акцент зроблено не лише на проходженні тестів, а й на гейміфікаційних механізмах, які підвищують залучення користувача: використання віртуальної валюти, нагород, рівнів, таблиць лідерів. На відміну від закритих екосистем, розроблений вебдодаток дозволяє легко додавати нові мови, теми, тести та досягнення шляхом редагування відповідних JSON-файлів у Firebase Storage.

Перспективи розвитку системи є широкими. У подальшій роботі доцільно реалізувати редактор тестів для адміністраторів або викладачів, що дозволить створювати й редагувати контент безпосередньо через інтерфейс платформи. Також перспективною є реалізація адаптивного тестування, що підбирає складність завдань відповідно до успішності користувача. Можливе підключення генеративного штучного інтелекту для автоматичного створення питань і варіантів відповідей, а також інтеграція з голосовими асистентами для розвитку навичок аудіювання.

З технічного боку можна передбачити імплементацію офлайн-режиму завдяки PWA-можливостям, що дозволить використовувати платформу в умовах обмеженого доступу до інтернету. Також доцільно реалізувати систему групового навчання, яка дозволить викладачам відстежувати прогрес учнів та створювати спільні рейтинги.

У підсумку, створена система є повноцінним MVP гейміфікованої платформи для мовного навчання, яка вже на поточному етапі здатна забезпечити індивідуальний підхід до кожного користувача, підвищити мотивацію до навчання та слугувати основою для подальшого розвитку як комерційного, так і освітнього проєкту. Вона є прикладом успішної інтеграції сучасних вебтехнологій з освітніми методиками, спрямованими на підвищення ефективності засвоєння іноземних мов.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бондар О. Ю. Гейміфікація в освіті як засіб активізації пізнавальної діяльності студентів. Інформаційні технології і засоби навчання. 2021. № 3(83). С. 45–55.

2. Гейміфікація в освіті: переваги, недоліки та приклади застосування. URL: https://osvitoria.media/opinions/gejmifikatsiya-v-osviti-perevagy-nedoliky-ta-pryklady-zastosuvannya (дата звернення: 12.02.2025).

3. Капранова С. Сучасні аспекти цифрової трансформації в освіті. *Освітній простір України.* 2022.

4. Платформи для вивчення іноземних мов: огляд та порівняння. URL: https://dou.ua/lenta/articles/language-learning-apps-comparison (дата звернення: 19.01.2025).

5. Розробка веб-застосунків: сучасні підходи та технології. URL: https://habr.com/ru/articles/738310 (дата звернення: 18.01.2025).

6. Babbel. URL: https://www.babbel.com (date of access: 04.04.2025).

7. Busuu. URL: https://www.busuu.com (date of access: 04.04.2025).

8. Clark R. C., Mayer R. E. E-learning and the science of instruction: Proven guidelines for consumers and designers of multimedia learning. Hoboken, NJ: Wiley, 2016. 528 p.

9. Duolingo. URL: https://www.duolingo.com (date of access: 04.04.2025).

10. Firebase Documentation. Google Developers. URL: https://firebase.google.com/docs (date of access: 07.02.2025).

11. Gamification in language learning: A systematic review. *British Journal of Educational Technology.* URL: https://bera-journals.onlinelibrary.wiley.com/doi/abs/10.1111/bjet.13042 (date of access: 07.03.2025).

12. Joy UI documentation. URL: https://mui.com/joy-ui/getting-started/ (date of access: 07.02.2025).

13. LingQ. URL: https://www.lingq.com (date of access: 04.04.2025).

14. Memrise. URL: https://www.memrise.com (date of access: 04.04.2025).

15. Mondly. URL: https://www.mondly.com (date of access: 04.04.2025).

16. NestJS Documentation. URL: https://docs.nestjs.com (date of access: 05.02.2025).

17. Pappas C. Top 20 Free Educational Apps For Students. URL: https://elearningindustry.com/top-20-free-educational-apps-for-students (date of access: 12.04.2025).

18. Racheva V. Gamification in e-learning. *International Journal of Cognitive Research in Science, Engineering and Education.* 2018. Vol. 6, No. 2. P. 81–88. URL: https://doi.org/10.5937/ijcrsee1802081R

19. Sailer M., Homner L. The gamification of learning: a meta-analysis. *Educational Psychology Review.* 2020. Vol. 32. P. 77–112. URL: https://doi.org/10.1007/s10648-019-09498-w

20. Zainuddin Z., Chu S. K. W., Shujahat M., Perera C. J. The impact of gamification on learning and instruction: A systematic review of empirical evidence. *Education and Information Technologies.* 2020. Vol. 25. P. 5161–5204. URL: https://doi.org/10.1007/s10639-020-10120-0

21. Zichermann G., Cunningham C. Gamification by design: Implementing game mechanics in web and mobile apps. Sebastopol, CA: O'Reilly Media, 2011. 208 p.