МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ
Циклова комісія комп’ютерної інженерії та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему

ВЕБ ЗАСТОСУНОК ДЛЯ ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ З ІНТЕРАКТИВНИМИ ВПРАВАМИ

Виконала: студентка групи 1П-22
Спеціальності
121 Інженерія програмного забезпечення
Валерія ДОРОШЕНКО
Керівник:
Вікторія НЕМЧЕНКО

Черкаси 2026

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ

Циклова комісія комп'ютерної інженерії та інформаційних технологій

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувачка циклової комісії КІ та ІТ

_____ Наталя ФАЛЬЧЕНКО

(підпис)

«_____» _____2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Дорошенко Валарії Миколаївні

1. Тема кваліфікаційної роботи «Розробка веб застосунку для вивчення англійської мови з інтерактивними вправами»

Керівник роботи Немченко Вікторія Юріївна, викладач другої категорії затверджені наказом закладу фахової передвищої освіти від «06» жовтня 2026 року №84/1-у

2. Строк подання студентом кваліфікаційної роботи 02.06.2026

3. Вихідні дані до кваліфікаційної роботи: мова програмування JavaScript, бібліотека React 19, бібліотека маршрутизації React Router, серверне середовище Node.js, фреймворк Express 5, база даних SQLite, сесійний механізм автентифікації, технології HTML, CSS, JSON, архітектурний стиль REST API, інтернаціоналізація інтерфейсу (i18n), середовище розробки Visual Studio Code.

4. Зміст кваліфікаційної роботи: огляд особливостей використання веб технологій у навчальному процесі; аналіз інтерактивних вправ як засобу вивчення англійської мови; аналіз існуючих веб застосунків для вивчення англійської мови; постановка задачі та аналіз вимог до програмного забезпечення; вибір технологій та інструментів розробки; проєктування архітектури вебзастосунку, моделі даних і користувацької взаємодії (UML); програмна реалізація каталогу вправ, авторизації, особистого кабінету та адміністративної панелі; тестування основних функціональних сценаріїв (функціональне, кросбраузерне та юзабіліті-тестування).

5. Дата видачі завдання 15.09.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	14.10.2025	
2	Розділ 1. Огляд поточного стану предметної області	09.12.2025	
3	Розділ 2. Проєктування та реалізація програмного проекту	10.03.2026	
4	Розділ 3. Тестування та супроводження	28.04.2026	
5	Висновки	12.05.2026	
6	Оформлення кваліфікаційної роботи (чистовий варіант)	26.05.2026	
7	Перевірка кваліфікаційної роботи на наявність ознак плагіату (за 10 днів до захисту)	02.06.2026	
8	Подання кваліфікаційної роботи на затвердження завідувачці циклової комісії (за 7 днів до захисту)	10.06.2026	

Студент _____ Валерія ДОРОШЕНКО
(підпис)

Керівник роботи_____Вікторія НЕМЧЕНКО
(підпис)

АНОТАЦІЯ

Кваліфікаційну роботу присвячено створенню веб застосунку для вивчення англійської мови з інтерактивними вправами. Окрему увагу приділено інтерактивним вправам – вони підвищують мотивацію того, хто навчається, і роблять засвоєння матеріалу відчутно результативнішим.

У межах роботи спроєктовано архітектуру застосунку, побудовано інтерфейс користувача та реалізовано логіку серверної частини. Клієнтську частину виконано як односторінковий застосунок (SPA) на бібліотеці React 19, серверну частину – на фреймворку Express 5 поверх платформи Node.js, а дані зберігаються у базі SQLite. Реалізовано автентифікацію на основі сесій із двома ролями (адміністратор і звичайний користувач), створення та проходження вправ, оцінювання їх, а також особистий кабінет, де можна завантажити аватар і перемкнути мову інтерфейсу між українською та англійською.

Сформовано бібліотеку вправ, що охоплюють шість часів англійської мови: Present Simple, Present Continuous, Present Perfect, Past Simple, Past Continuous та Future Simple. Кожна вправа завдання трьох типів – тест із вибором відповіді, заповнення пропуску та встановлення відповідностей. Підсумковий бал обчислюється автоматично. Коректність роботи всіх модулів підтверджено функціональним, кросбраузерним та юзабіліті-тестуванням.

Готовий застосунок придатний як для самостійного вивчення англійської, так і для використання викладачем у ролі допоміжного інструмента в закладі освіти.

Ключові слова: ВЕБ ЗАСТОСУНОК, АНГЛІЙСЬКА МОВА, ІНТЕРАКТИВНІ ВПРАВИ, REACT, NODE.JS, EXPRESS, SQLITE, СЕСІЙНА АВТЕНТИФІКАЦІЯ, ОДНОСТОРІНКОВИЙ ЗАСТОСУНОК.

ANNOTATION

The qualification work is devoted to the creation of a web application for learning English with interactive exercises. Special attention is paid to interactive exercises – they increase the motivation of the learner and make the assimilation of the material noticeably more effective.

Within the scope of the work, the application architecture was designed, the user interface was built and the logic of the server side was implemented. The client part is implemented as a single-page application (SPA) based on the React 19 library, and the server part – on the Express 5 framework on top of the Node.js platform, while the data is stored in an SQLite database. Session-based authentication with two roles (administrator and regular user) was implemented, as well as the creation and completion of exercises, their rating, and a personal cabinet where one can upload an avatar and switch the interface language between Ukrainian and English.

A library of exercises was formed, covering six English tenses: Present Simple, Present Continuous, Present Perfect, Past Simple, Past Continuous and Future Simple. Each exercise contains tasks of three types – a multiple-choice test, gap filling and matching. The final score is calculated automatically. The correct operation of all modules was confirmed by functional, cross-browser and usability testing.

The finished application is suitable both for independent study of English and for use by a teacher as an auxiliary tool in an educational institution.

Keywords: WEB APPLICATION, ENGLISH LANGUAGE, INTERACTIVE EXERCISES, REACT, NODE.JS, EXPRESS, SQLITE, SESSION AUTHENTICATION, SINGLE-PAGE APPLICATION.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ 3

ВСТУП 4

РОЗДІЛ 1 ОГЛЯД ПОТОЧНОГО СТАНУ ПРЕДМЕТНОЇ ОБЛАСТІ 7

1.1 Особливості використання веб технологій у навчальному процесі 7

1.2 Інтерактивні вправи як засіб вивчення англійської мови 8

1.3. Аналіз існуючих веб застосунків для вивчення англійської мови 9

1.4 Обґрунтування вибору підходу до розробки веб застосунку 12

РОЗДІЛ 2 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОЕКТУ 14

2.1 Аналіз вимог до програмного забезпечення, що розробляється 14

2.2 Вибір технологій та інструментів розробки 15

2.3 Проектування архітектури веб застосунку 16

2.4 Проектування моделі даних 18

2.5 Моделювання користувацької взаємодії (UML) 23

2.6 Програмна реалізація проекту 26

РОЗДІЛ 3 ТЕСТУВАННЯ ТА СУПРОВОДЖЕННЯ 33

3.1 Перелік і обґрунтування обраних методів тестування 33

3.2 Тестовий план проекту 34

3.3 Аналіз отриманих результатів 36

ВИСНОВКИ 39

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 41

ДОДАТКИ 44

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ

HTML (HyperText Markup Language)	Мова гіпертекстової розмітки
CSS (Cascading Style Sheets)	Каскадні таблиці стилів
JS (JavaScript)	Мова програмування для реалізації інтерактивності веб сторінок
JSX (JavaScript XML)	Розширення синтаксису JavaScript для опису UI у React
SPA (Single Page Application)	Односторінковий веб застосунок
API (Application Programming Interface)	Програмний інтерфейс взаємодії компонентів
REST (Representational State Transfer)	Архітектурний стиль для побудови мережевих API
HTTP (HyperText Transfer Protocol)	Протокол передавання гіпертексту
JSON (JavaScript Object Notation)	Текстовий формат обміну даними
SQL (Structured Query Language)	Структурована мова запитів до баз даних
DOM (Document Object Model)	Об’єктна модель документа
UI (User Interface)	Користувацький інтерфейс
UX (User Experience)	Користувацький досвід
i18n (Internationalization)	Інтернаціоналізація користувацького інтерфейсу
ORM (Object-Relational Mapping)	Об’єктно-реляційне відображення
CRUD (Create, Read, Update, Delete)	Базові операції роботи з даними

ВСТУП

Сьогодні, в умовах глобалізації, знання англійської мови стало однією з опор і професійного, і особистого розвитку. Саме англійська виконує роль міжнародної мови спілкування в науці, бізнесі та інформаційних технологіях. Проте традиційні методи навчання не завжди вдається поєднати з достатнім рівнем зацікавленості й активної участі здобувачів освіти. Через це дедалі помітнішою стає потреба в інформаційних технологіях – передусім у веб застосунках, які дають змогу організувати навчання у зручному та інтерактивному форматі.

Онлайн-ресурсів і платформ для вивчення англійської нині справді багато. Більшість із них – комерційні сервіси, нерідко перевантажені функціоналом або розраховані на певну категорію користувачів. Це ускладнює їх пристосування до навчальних цілей конкретного закладу освіти. Тому актуальним залишається створення власного навчального веб застосунку, який забезпечував би опанування мови через інтерактивні вправи, зосереджені на тренуванні англійських часів – однієї з найскладніших граматичних тем для українськомовних користувачів.

Актуальність теми зумовлена потребою у навчальних веб застосунках, що поєднують зручний інтерфейс, інтерактивність, збереження прогресу та можливість розширення матеріалу адміністратором. Сучасні клієнт-серверні технології дозволяють будувати ефективні інструменти навчання, доступні широкому колу користувачів через звичайний браузер.

Об’єктом дослідження є система вивчення англійської мови з використанням інтерактивних веб технологій.

Предметом дослідження є методи, засоби та технології розробки клієнт-серверного веб застосунку з інтерактивними вправами для вивчення часів в англійській мові, зокрема проєктування архітектури системи, реалізація

користувацького інтерфейсу та серверної логіки, організація взаємодії між клієнтом і сервером, робота з базою даних, система автентифікації, механізми перевірки й оцінювання знань, а також забезпечення зручності, адаптивності та ефективності навчального процесу.

Метою дипломної роботи є розробка веб застосунку для вивчення англійської мови з інтерактивними вправами, який реалізує сесійну автентифікацію, розподіл користувацьких ролей, оцінювання результатів та можливість розширення бази вправ адміністратором.

Для досягнення поставленої мети потрібно вирішити такі завдання:

1. проаналізувати сучасні веб застосунки для вивчення англійської мови та виявити їхні переваги і недоліки;
2. визначити функціональні та нефункціональні вимоги до навчального веб застосунку;
3. спроєктувати архітектуру клієнт-серверного веб застосунку та обрати стек технологій;
4. спроєктувати модель даних для збереження користувачів, вправ, питань, результатів та оцінок;
5. реалізувати клієнтську частину у вигляді SPA з підтримкою маршрутизації та переключення мови інтерфейсу;
6. реалізувати серверну частину з REST API, сесійною автентифікацією та контролем доступу за ролями;
7. наповнити базу даних інтерактивними вправами для шести часів англійської мови;
8. реалізувати оцінювання правильності виконання вправ, систему рейтингу та особистий кабінет користувача;
9. провести функціональне, кросбраузерне та юзабіліті-тестування розробленого застосунку.

Практичне значення роботи полягає у створенні готового веб застосунку, який можуть використовувати студенти й викладачі для самостійного вивчення англійської мови та як допоміжний інструмент у навчальному процесі. Адміністратор здатен самостійно розширювати бібліотеку вправ, тож застосунок масштабується без участі розробника.

РОЗДІЛ 1 ОГЛЯД ПОТОЧНОГО СТАНУ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Особливості використання веб технологій у навчальному процесі

Стрімкий розвиток інформаційних технологій помітно змінив підходи до організації навчання. Застосування веб технологій робить матеріали доступними, дає свободу у виборі часу й темпу занять і водночас підвищує зацікавленість користувачів. Особливо дієвими виявляються веб застосунки, у яких теоретичний матеріал поєднано з практичними інтерактивними завданнями та збереженням прогресу.

Порівняно з традиційними програмами веб застосунки мають кілька вагомих переваг: їх не потрібно встановлювати, вони крос-платформні й відкриваються з будь-якого пристрою із сучасним браузером. Сучасний підхід до розробки передбачає архітектуру односторінкового застосунку (SPA), коли увесь інтерфейс формується на боці клієнта, а обмін із сервером відбувається через асинхронні запити до REST API. Завдяки цьому інтерфейс працює плавно й швидко, без перезавантаження сторінок.

Ще одна важлива риса навчальних веб застосунків – інтерактивність, яка забезпечує активну взаємодію користувача з матеріалом. Інтерактивні елементи дозволяють одразу перевіряти знання, отримувати миттєвий зворотний зв’язок і коригувати процес навчання. Збереження результатів у базі даних, своєю чергою, дає змогу відстежувати динаміку засвоєння та вибудовувати індивідуальну освітню траєкторію.

1.2 Інтерактивні вправи як засіб вивчення англійської мови

Інтерактивні вправи є дієвим інструментом формування мовних навичок, адже залучають користувача до навчання активно, а не пасивно. На відміну від простого сприйняття інформації, виконання таких завдань змушує працювати мислення, увагу й пам’ять. Для вивчення англійських часів цей формат особливо доречний: система часів складна для україномовних здобувачів освіти й потребує регулярної практики в різних контекстах.

Серед найпоширеніших типів інтерактивних вправ для вивчення англійської мови можна виокремити такі:

– тести з вибором правильної відповіді (multiple choice);

– вправи на заповнення пропусків у реченні (gap filling);

– завдання на встановлення відповідності між словами та значеннями (matching);

– вправи на переклад слів і речень;

– аудіо- та відеовправи на сприйняття мови на слух.

Такі завдання тренують різні складники мовної компетентності – лексику, граматику й розуміння контексту. Миттєвий зворотний зв’язок, який дають інтерактивні елементи, допомагає користувачеві самому контролювати рівень засвоєння. Якщо ж застосовувати такі вправи систематично, зростає мотивація і формується позитивний користувацький досвід.

1.3. Аналіз існуючих веб застосунків для вивчення англійської мови

На ринку представлено чимало веб застосунків та онлайн-платформ для вивчення англійської мови. До найвідоміших належать Duolingo, Memrise і BBC Learning English. Розгляньмо їхні особливості докладніше.

1.3.1. Duolingo

Duolingo пропонує навчання у форматі коротких інтерактивних вправ із елементами гейміфікації. Сильні сторони сервісу – привабливий інтерфейс, висока мотивація завдяки системі щоденних серій (streaks) і підтримка великої кількості мовних пар. Проте сервіс орієнтований на загальний курс і не дозволяє викладачеві чи користувачеві пристосувати вправи під власні потреби. Безкоштовна версія містить рекламу та обмежену кількість «життів», що змушує користувачів або чекати, або переходити на платний тариф.

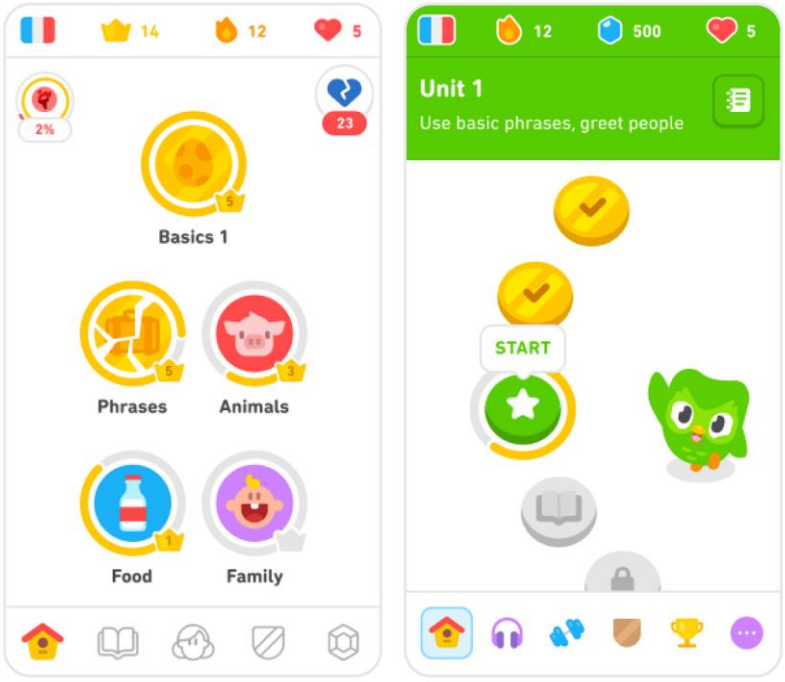

Рисунок 1.1 – Duolingo

1.3.2 Memrise

Memrise зосереджено передусім на запам’ятовуванні лексики за допомогою мультимедійного контенту, зокрема відео з носіями мови. Інтерфейс зручний, однак більшість розширених можливостей – офлайн-режим, аналітика прогресу – доступні лише у платній версії Pro. Граматичні вправи на часи представлені обмежено, тож для системного опанування граматики Memrise підходить не надто добре.

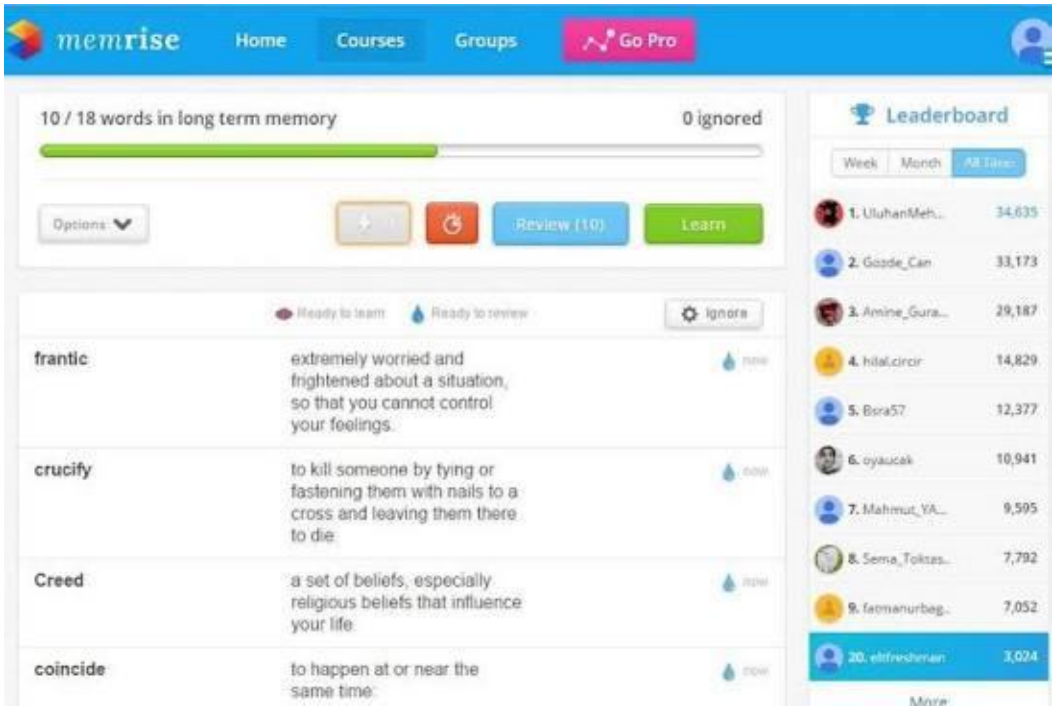

Рисунок 1.2 – Memrice

1.3.3 BBC Learning English

BBC Learning English дає якісний навчальний матеріал, орієнтований на поглиблене вивчення мови через статті, відео й аудіо. Водночас ресурс менш інтерактивний: значна частина контенту подана пасивно, без автоматичної перевірки відповідей і без збереження прогресу.

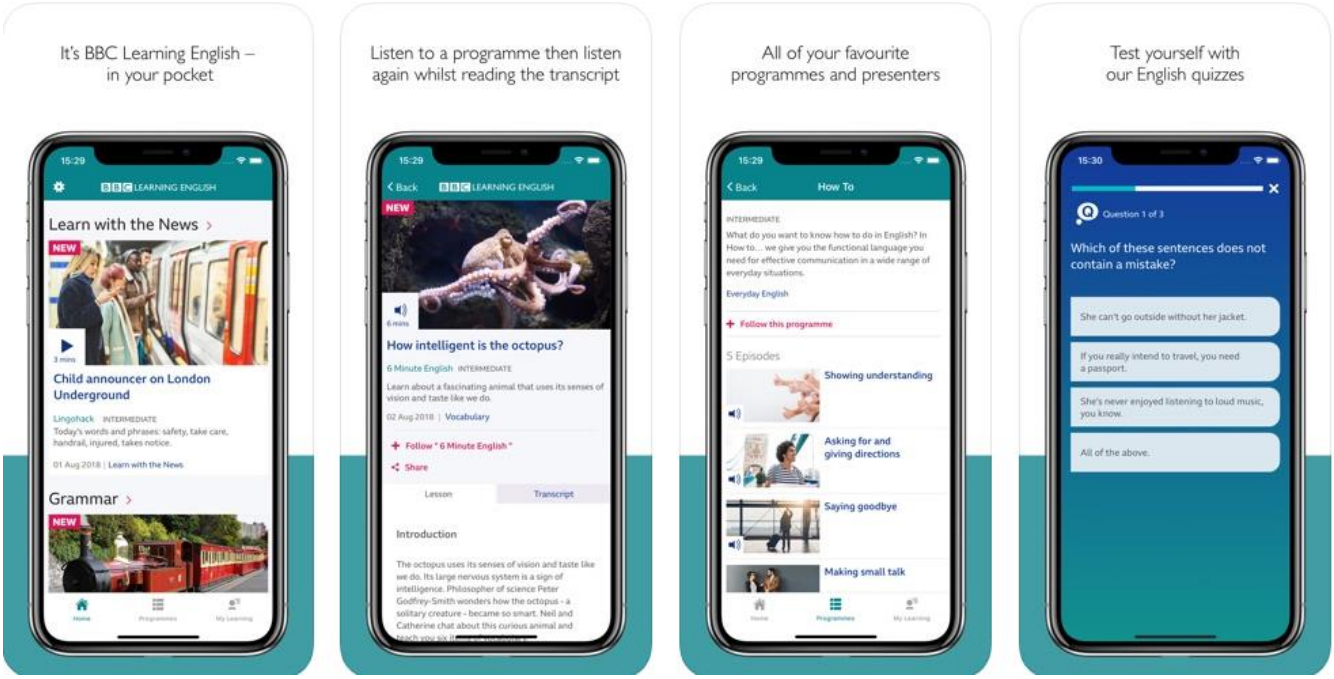

Рисунок 1.3 – BBC Learning English

Порівняльну характеристику розглянутих платформ наведено у табл. 1.1.

Таблиця 1.1 – Порівняльна характеристика платформ для вивчення англійської мови

Критерій	Duolingo	Memrise	BBC Learning English
Інтерактивні тести	Так	Так	Частково
Вправи на пропуски	Так	Обмежено	Ні
Вправи на відповідність	Так	Так	Ні
Збереження прогресу	Так	Так	Ні
Адаптація викладачем	Ні	Ні	Ні
Сортування за часами	Частково	Ні	Частково
Безкоштовність	З обмеженнями	З обмеженнями	Так
Українська локалізація	Так	Частково	Ні

1.4 Обґрунтування вибору підходу до розробки веб застосунку

Проведений аналіз показує, що більшість популярних платформ обмежено гнучкі у налаштуванні навчального матеріалу й розраховані на широку аудиторію. Це заважає використовувати їх у конкретному закладі освіти, де викладач хотів би самостійно формувати набір вправ під поточні потреби курсу.

З огляду на виявлені недоліки ухвалено рішення розробити власний веб застосунок із такими ключовими особливостями:

– сортування вправ за часами англійської мови, щоб цілеспрямовано тренувати конкретний граматичний матеріал;

– сесійна автентифікація та особистий кабінет, що зберігає історію проходження вправ і результати;

– роль адміністратора з можливістю додавати нові вправи без участі розробника;

– система рейтингу вправ у п’ять зірок для оцінювання якості матеріалу спільнотою користувачів;

– підтримка двох мов інтерфейсу (української та англійської) для ширшої аудиторії;

– повна безкоштовність та відсутність комерційних обмежень.

Для реалізації обрано клієнт-серверну архітектуру: клієнт – у вигляді односторінкового застосунку (SPA), серверна частина – у вигляді REST API, а дані зберігаються у реляційній базі. Обґрунтування конкретного стеку технологій наведено у розділі 2.

Висновки до розділу 1

У розділі розглянуто роль веб технологій у навчанні та переваги інтерактивних вправ для опанування англійських часів. Аналіз трьох популярних платформ – Duolingo, Memrise і BBC Learning English – показав їхню спільну ваду: вони не дозволяють викладачеві самостійно формувати й адаптувати навчальний матеріал. Це й стало підставою для рішення розробити власний застосунок із сортуванням вправ за часами, роллю адміністратора, збереженням прогресу та двомовним інтерфейсом. Сформульовані тут висновки лягли в основу вимог, які детально розглянуто у наступному розділі.

РОЗДІЛ 2 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОЕКТУ

2.1 Аналіз вимог до програмного забезпечення, що розробляється

На основі аналізу предметної області та програмних аналогів сформовано перелік вимог до веб застосунку для вивчення англійської мови. Вимоги поділено на функціональні та нефункціональні.

Функціональні вимоги:

– реєстрація та автентифікація користувачів із використанням сесій;

– розподіл користувацьких ролей: адміністратор та звичайний користувач;

– створення та редагування вправ адміністратором через окремий інтерфейс;

– сортування та фільтрація вправ за часами англійської мови;

– проходження вправ із 20 завдань трьох типів: тести з вибором відповіді (10 шт.), заповнення пропусків (5 шт.), встановлення відповідностей (5 шт.);

– автоматичне оцінювання правильності відповідей із максимальним результатом 10 балів (по 0,5 бала за правильну відповідь);

– оцінювання вправ користувачами у системі рейтингу 5 зірок;

– особистий кабінет (налаштування профілю) із можливістю змінити аватар та переглянути історію проходження вправ;

– переключення мови інтерфейсу між українською та англійською без перезавантаження сторінки.

Нефункціональні вимоги:

– інтуїтивно зрозумілий інтерфейс із підтримкою світлої та темної теми;

– коректна робота у сучасних браузерах (Chrome, Firefox, Edge);

– адаптивність до різних розмірів екранів (десктоп, планшет, мобільний пристрій);

– захист паролів за допомогою криптографічного хешування (bcrypt);

– відсутність потреби встановлювати додаткове програмне забезпечення на боці користувача;

– можливість локального розгортання застосунку для навчальних цілей.

– Описані вимоги забезпечують цілісну функціональність застосунку та слугують основою для подальшого проєктування його архітектури.

2.2 Вибір технологій та інструментів розробки

Для застосунку обрано клієнт-серверну архітектуру з REST-комунікацією між фронтендом і бекендом. Стек технологій підібрано з огляду на сучасні стандарти веб розробки, активну спільноту та власний досвід.

Технології клієнтської частини:

– React 19 – бібліотека для побудови компонентного інтерфейсу;

– React Router 7 – бібліотека для клієнтської маршрутизації між сторінками SPA;

– Vite 8 – інструмент збирання та локальний сервер розробки з гарячим перезавантаженням модулів (HMR);

– JSX та сучасний JavaScript (ES2024) для опису UI;

– CSS Custom Properties – для реалізації світлої й темної теми та дизайн-токенів.

Технології серверної частини:

– Node.js – серверне середовище виконання JavaScript;

– Express 5 – мінімалістичний веб фреймворк для побудови REST API;

– express-session – бібліотека для роботи з HTTP-сесіями;

– bcrypt – бібліотека для криптографічного хешування паролів;

– multer – бібліотека для обробки multipart-завантаження файлів (аватарів);

– CORS – middleware для налаштування політик безпеки між доменами.

– Зберігання даних:

– SQLite 3 – компактна вбудована реляційна СУБД, що не потребує окремого процесу-сервера й добре пасує до навчальних застосунків та невеликих навантажень.

Обґрунтування вибору. React обрано через його поширеність і компонентний підхід, який дає змогу повторно використовувати елементи інтерфейсу. Express має низький поріг входження та дозволяє швидко описати REST API. SQLite не вимагає окремого сервера баз даних, що спрощує локальне розгортання. Сесійну автентифікацію обрано замість токенної (JWT) як простішу й доречнішу для застосунків із серверним станом, де клієнт і сервер працюють у межах одних доменів і потрібен миттєвий вихід із системи (інвалідація сесії на сервері).

2.3 Проектування архітектури веб застосунку

Застосунок складається з двох основних частин: клієнтської (SPA-фронтенд) та серверної (REST API і база даних). Загальну схему взаємодії наведено у табл. 2.1.

Клієнтську частину побудовано як SPA: усі переходи між сторінками (головна, авторизація, реєстрація, список вправ, проходження вправи, особистий кабінет, адмін-панель) відбуваються без перезавантаження сторінки. Маршрутизацію реалізовано через React Router. Стан автентифікації зберігається у

React-контексті AuthContext, а поточна мова інтерфейсу – у контексті I18nContext із синхронізацією у localStorage.

Серверну частину організовано за модульним принципом: окремі файли маршрутів для автентифікації, реєстрації, профілю та вправ; окремий модуль ініціалізації схеми бази даних; окремий модуль middleware для перевірки автентифікації та ролі адміністратора. Уся серверна логіка написана на Node.js без додаткового шару ORM – запити до бази виконуються напряму через драйвер sqlite3 з параметризацією, що захищає від SQL-ін’єкцій.

Усі захищені маршрути API перевіряють наявність активної сесії через middleware requireAuth, а адміністративні маршрути додатково вимагають роль admin через middleware requireAdmin.

Таблиця 2.1 – Рівні архітектури веб застосунку

Рівень	Технології	Відповідальність
Клієнт	React + React Router + CSS	Маршрутизація сторінок, рендеринг компонентів UI, локальне зберігання мови, форми та валідація на стороні клієнта
Транспорт	HTTP + JSON + cookies	Передавання запитів між клієнтом і сервером, підтримка сесійного cookie connect.sid
Сервер (API)	Node.js + Express	Маршрути REST API, перевірка автентифікації та ролей, бізнес-логіка оцінювання, рейтингу, завантаження файлів
Зберігання даних	SQLite	Збереження користувачів, вправ, питань, результатів проходження, оцінок рейтингу

2.4 Проектування моделі даних

Модель даних застосунку охоплює п’ять основних сутностей: користувачі, вправи, питання, результати проходження та оцінки. Кожній сутності відповідає окрема таблиця бази даних SQLite, тож загалом схема містить п’ять таблиць. Стисле призначення кожної таблиці наведено у табл. 2.2, а повний перелік полів із типами та поясненнями – у табл. 2.3.

Таблиця 2.2 – Перелік таблиць бази даних та їх призначення

Таблиця	Призначення
users	Облікові записи користувачів: логін, e-mail, хеш пароля, роль (user / admin) та шлях до файлу аватара
exercises	Інтерактивні вправи з прив’язкою до часу англійської мови та автора-адміністратора
questions	Окремі питання вправ трьох типів – test, fill, match – із варіантами та правильною відповіддю
results	Історія проходження вправ із обчисленим балом та кількістю правильних відповідей
ratings	Оцінки вправ користувачами за 5-зірковою шкалою з обмеженням одна оцінка на користувача

Таблиця users зберігає облікові записи: первинний ключ id, унікальні поля login та email, хеш пароля password (алгоритм bcrypt), роль role зі значенням за замовчуванням user, необов’язкове поле avatar зі шляхом до завантаженого зображення та позначку часу створення created_at. Таблиця exercises описує вправу: її назву title, час англійської мови tense, опис description, посилання на автора created_by та дату створення. Таблиця questions містить питання вправ;

поле type визначає тип питання (test, fill або match), position задає порядок, prompt зберігає текст завдання, options – серіалізований JSON-масив варіантів (для типів test і match) або NULL (для типу fill), а answer – правильну відповідь. Таблиця results фіксує кожне проходження: бал score, кількість правильних відповідей correct_count, загальну кількість питань total_count та час завершення. Таблиця ratings зберігає оцінки: кількість зірок stars (від 1 до 5) із обмеженням цілісності та унікальною парою user_id + exercise_id.

Повний опис полів усіх таблиць наведено у табл. 2.3.

Таблиця 2.3 – Детальна структура полів таблиць бази даних

Таблиця	Поле	Тип	Опис
1	2	3	4
users	id	INTEGER PK	Унікальний ідентифікатор користувача (автоінкремент)
users	login	TEXT UNIQUE	Логін користувача
users	email	TEXT UNIQUE	Адреса електронної пошти
users	password	TEXT	Хеш пароля, обчислений алгоритмом bcrypt
users	role	TEXT	Роль: user або admin (за замовчуванням user)
users	avatar	TEXT	Шлях до файлу аватара (може бути порожнім)
users	created_at	DATETIME	Дата й час реєстрації
exercises	id	INTEGER PK	Ідентифікатор вправи
exercises	title	TEXT	Назва вправи

Продовження таблиці 2.3

1	2	3	4
exercises	tense	TEXT	Час англійської мови, до якого належить вправа
exercises	description	TEXT	Короткий опис вправи
exercises	created_by	INTEGER FK	Автор вправи (посилання на users.id)
exercises	created_at	DATETIME	Дата й час створення
questions	id	INTEGER PK	Ідентифікатор питання
questions	exercise_id	INTEGER FK	Вправа, до якої належить питання (ON DELETE CASCADE)
questions	type	TEXT	Тип питання: test, fill або match
questions	position	INTEGER	Порядковий номер питання у вправі
questions	prompt	TEXT	Текст завдання
questions	options	TEXT	JSON-масив варіантів (test, match) або NULL (fill)
questions	answer	TEXT	Правильна відповідь
results	id	INTEGER PK	Ідентифікатор результату
results	user_id	INTEGER FK	Користувач (посилання на users.id)
results	exercise_id	INTEGER FK	Вправа (посилання на exercises.id)
results	score	REAL	Підсумковий бал (максимум 10)
results	correct_count	INTEGER	Кількість правильних відповідей
results	total_count	INTEGER	Загальна кількість питань

Продовження таблиці 2.3

1	2	3	4
results	finished_at	DATETIME	Час завершення проходження
ratings	id	INTEGER PK	Ідентифікатор оцінки
ratings	user_id	INTEGER FK	Користувач, який оцінив вправу
ratings	exercise_id	INTEGER FK	Оцінена вправа
ratings	stars	INTEGER	Оцінка від 1 до 5 (CHECK 1..5)
ratings	created_at	DATETIME	Дата й час оцінювання

Зв’язки між таблицями реалізовано через зовнішні ключі: exercises.created_by посилається на users.id; questions.exercise_id, results.exercise_id і ratings.exercise_id посилаються на exercises.id; results.user_id і ratings.user_id посилаються на users.id. На таблицю ratings накладено обмеження UNIQUE(user_id, exercise_id) – воно гарантує одну оцінку від користувача на одну вправу, а повторне оцінювання оновлює попереднє значення. Завдяки тому, що всі три типи питань зберігаються в одній таблиці questions (а поле options для типу fill дорівнює NULL), схема залишається простою й не потребує окремих таблиць під кожен тип питання.

Узагальнену структуру бази даних та зв’язки між сутностями наведено на ER-діаграмі (рис. 2.1).

Нормалізація бази даних. Базу даних спроєктовано відповідно до принципів реляційної нормалізації, і її схема перебуває у третій нормальній формі (3НФ).

Перша нормальна форма (1НФ): кожна таблиця має первинний ключ, а її атрибути зберігають атомарні значення — одне поле містить одне неподільне

значення, повторювані групи відсутні. Свідомим винятком є поле options таблиці questions, де перелік варіантів відповіді зберігається як серіалізований JSON-рядок: варіанти завжди використовуються разом із питанням і не потребують окремих запитів, що спрощує модель.

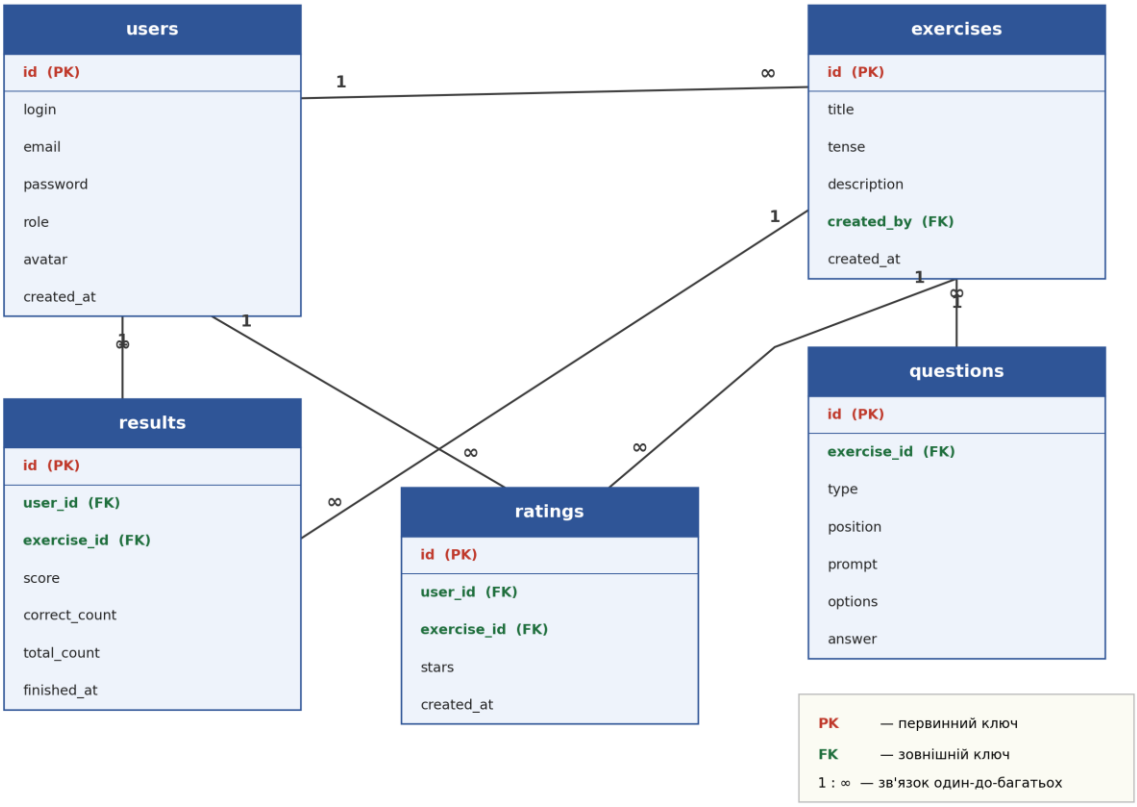

Рисунок 2.1 – ER-діаграма бази даних

Друга нормальна форма (2НФ): усі таблиці задовольняють 2НФ, оскільки первинний ключ у кожній з них простий — сурогатне поле id. За відсутності складених ключів часткові залежності неключових атрибутів від частини ключа неможливі.

Третя нормальна форма (3НФ): транзитивні залежності між неключовими атрибутами відсутні — усі неключові поля залежать лише від первинного ключа. Похідні дані не дублюються: середній рейтинг вправи не зберігається окремим

полем, а обчислюється з таблиці ratings під час запиту, тоді як поля score, correct_count і total_count таблиці results фіксують підсумок конкретного проходження вправи.

2.5 Моделювання користувацької взаємодії (UML)

Щоб дослідити взаємодію користувача із застосунком, основні сценарії змодельовано у вигляді UML-діаграми прецедентів (use-case). Виокремлено двох акторів – звичайного користувача та адміністратора. Діаграму прецедентів, побудовану засобом draw.io, наведено на рис. 2.2.

Рисунок 2.2 – Діаграма прецедентів (use-case) веб застосунку

Більшість прецедентів доступні обом ролям, тоді як реєстрація стосується лише звичайного користувача, а створення й видалення вправ – лише адміністратора. Функції зміни аватара та перегляду історії результатів об’єднано в єдиний прецедент «Налаштування профілю». Окремо варто відзначити зв’язок «extend» між прецедентами «Проходження вправи» та «Оцінювання вправи»: оцінити вправу можна лише після її проходження, тобто оцінювання розширює базовий сценарій проходження як необов’язкове продовження. Розподіл прецедентів за ролями наведено у табл. 2.4.

Динаміку взаємодії компонентів системи показано на діаграмі послідовності (рис. 2.3), яка ілюструє повний сценарій проходження вправи звичайним користувачем – від відкриття вправи до її оцінювання. На діаграмі задіяно чотирьох учасників: користувача, браузер (SPA), сервер (Express API) та базу даних SQLite.

Таблиця 2.4 – Розподіл прецедентів за ролями користувачів

Прецедент	Звичайний користувач	Адміністратор
Реєстрація	Так	–
Вхід / Вихід	Так	Так
Перегляд списку вправ	Так	Так
Проходження вправи	Так	Так
Оцінювання вправи	Так	Так
Перегляд результату	Так	Так
Налаштування профілю	Так	Так
Створення нової вправи	–	Так
Видалення вправи	–	Так

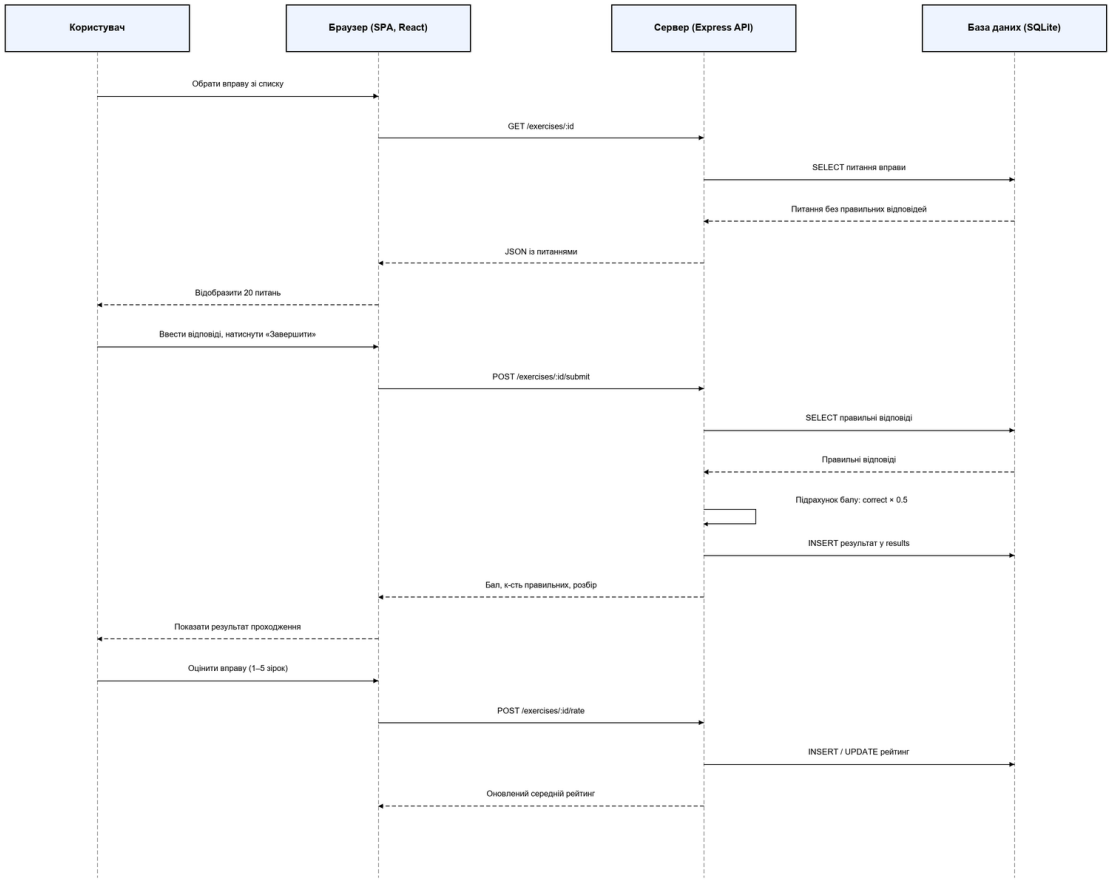

Рисунок 2.3 – Діаграма послідовності проходження та оцінювання вправи

Типовий сценарій проходження вправи звичайним користувачем складається з таких кроків:

– користувач переходить на сторінку списку вправ та обирає фільтр за часом;

– користувач відкриває вправу та послідовно відповідає на 20 питань трьох типів;

– по завершенні користувач натискає кнопку «Завершити»;

– сервер обчислює бал, зберігає результат у таблицю results і повертає клієнту бал, кількість правильних відповідей та правильні відповіді на помилкові питання;

– клієнт відображає результат із розбором кожного питання;

– користувач може оцінити вправу від 1 до 5 зірок (оновлюється середній рейтинг вправи).

– Сценарій створення вправи адміністратором:

– адміністратор переходить на сторінку «Адмін»;

– заповнює форму з назвою, часом та описом вправи;

– додає потрібну кількість питань кожного типу з варіантами та правильною відповіддю;

– сервер створює запис у таблиці exercises та пов’язані записи у questions;

– нова вправа стає доступною всім користувачам у списку.

2.6 Програмна реалізація проекту

Структуру каталогів проєкту наведено у табл. 2.5.

Таблиця 2.5 – Структура каталогів проєкту

Шлях	Призначення
1	2
src/	Клієнтська частина (React)
src/main.jsx	Точка входу: React Router, провайдери AuthContext і I18nContext
src/App.jsx	Опис маршрутів SPA та компонентів обмеження доступу
src/api.js	Обгортка fetch для звернень до REST API
src/auth.jsx	React-контекст автентифікації
src/i18n.jsx	React-контекст інтернаціоналізації UA/EN
src/components/	Спільні компоненти (Layout, Stars)

Продовження таблиц 2.5 – Структура каталогів проєкту

1	2
src/pages/	Сторінки SPA (Home, Login, Register, Exercises, Exercise, Profile, Admin)
src/index.css	Дизайн-система та глобальні стилі
server/	Серверна частина (Node.js + Express)
server/server.js	Точка входу сервера, налаштування middleware і маршрутів
server/db.js	Підключення до SQLite та ініціалізація схеми
server/seed.js	Скрипт наповнення бази тестовими даними
server/content.js	База інтерактивних питань за часами
server/middleware/auth.js	Middleware requireAuth, requireAdmin
server/routes/	Маршрути REST API: auth, register, profile, exercises
server/uploads/avatars/	Каталог для завантажених файлів аватарів

REST API застосунку реалізує набір ендпоінтів, перелік яких наведено у табл. 2.6.

Таблиця 2.6 – Ендпоінти REST API

HTTP метод	Шлях	Призначення
POST	/register	Реєстрація нового користувача
POST	/auth/login	Вхід у систему (створення сесії)
POST	/auth/logout	Вихід (знищення сесії)
GET	/auth/me	Отримання поточного користувача
GET	/exercises	Список вправ із середнім рейтингом

Продовження таблиці 2.6

1	2	3
GET	/exercises/:id	Вправа з питаннями (без правильних відповідей)
POST	/exercises/:id/submit	Надсилання відповідей та оцінювання
POST	/exercises/:id/rate	Оцінювання вправи 1–5 зірками
POST	/exercises	Створення вправи (admin)
DELETE	/exercises/:id	Видалення вправи (admin)
POST	/profile/avatar	Завантаження аватара
GET	/profile/results	Історія результатів користувача

Алгоритм оцінювання реалізовано на сервері у маршруті POST /exercises/:id/submit. Послідовність дій така:

– сервер отримує об’єкт answers у форматі { questionId: answer };

– із бази завантажуються правильні відповіді для всіх 20 питань вправи;

– для кожного питання виконується нормалізація рядка (trim, lowercase) та порівняння;

– підраховується кількість правильних відповідей correct;

– обчислюється підсумковий бал за формулою score = correct × 0,5, тобто максимум 10 балів;

– результат зберігається у таблицю results із прив’язкою до користувача та вправи;

– клієнту повертається об’єкт із балом, кількістю правильних відповідей та розбором кожного питання.

Інтернаціоналізацію реалізовано через власний контекст I18nContext із двома словниками перекладів (uk, en). Поточна мова зберігається у localStorage й відновлюється при наступних відвідуваннях, а перемикання відбувається миттєво, без перезавантаження сторінки.

Завантаження аватарів виконано за допомогою бібліотеки multer з обмеженням розміру файлу 2 МБ та фільтром за розширенням (png, jpg, jpeg, gif, webp). Файли зберігаються у каталозі server/uploads/avatars/ під унікальним ім’ям виду avatar_<userId>_<timestamp>.<ext>, а посилання на файл записується у поле users.avatar.

На рисунках 2.4, 2.5, 2.6, 2.7, 2.8, 2.9 зображено практичну частину проєкту.

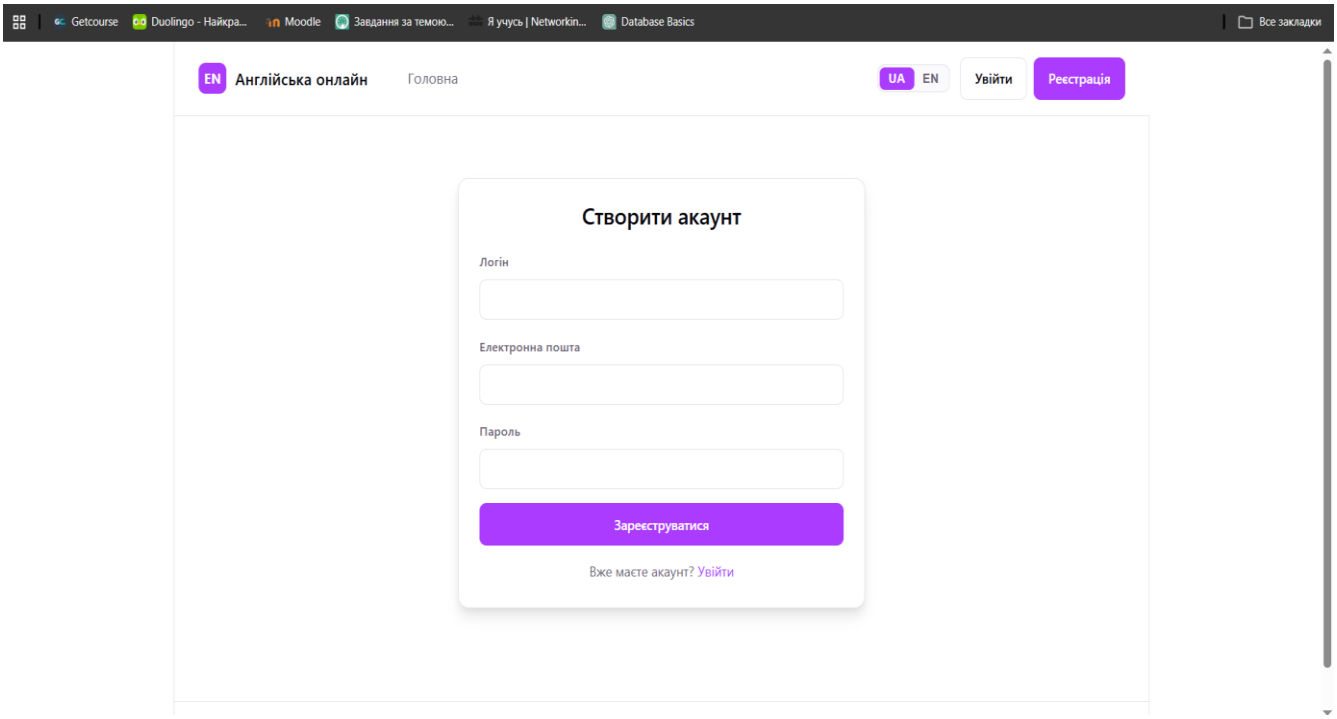

Рисунок 2.4 – Головна сторінка

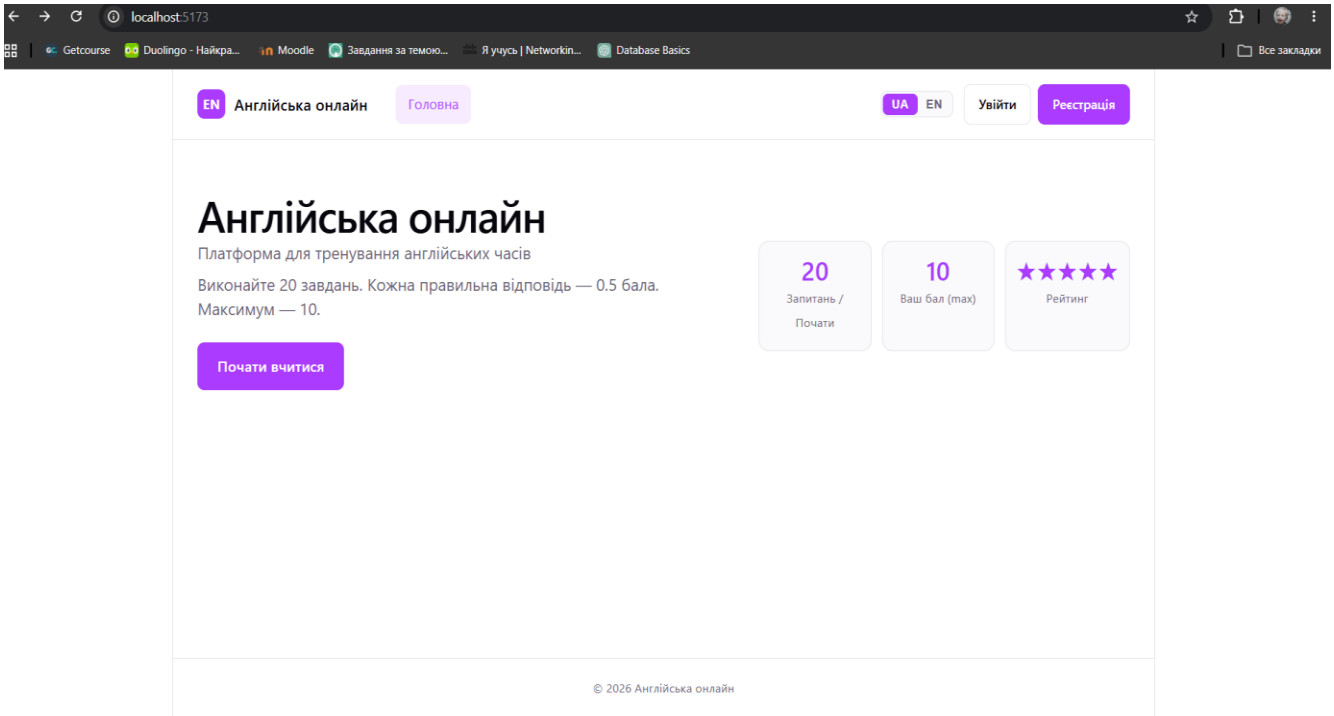

Рисунок 2.5 – Сторінка реєстрації

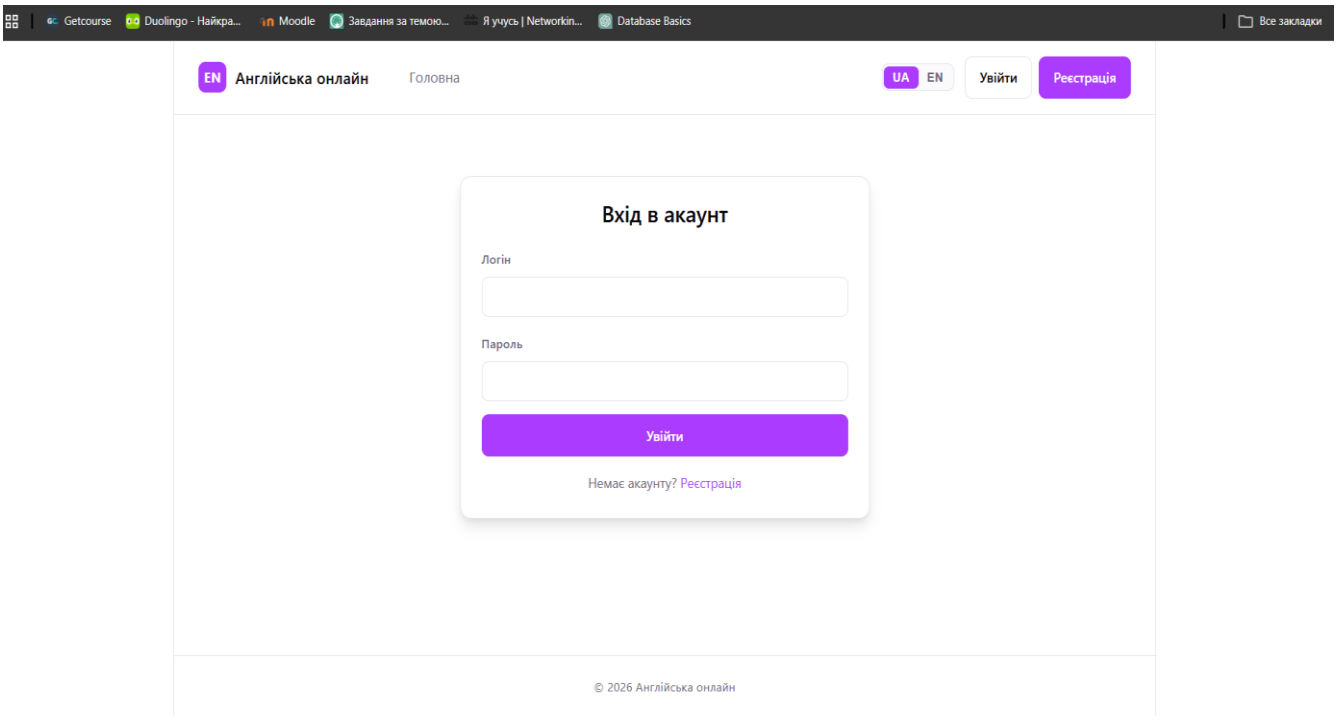

Рисунок 2.6 – Сторінка автентифікації

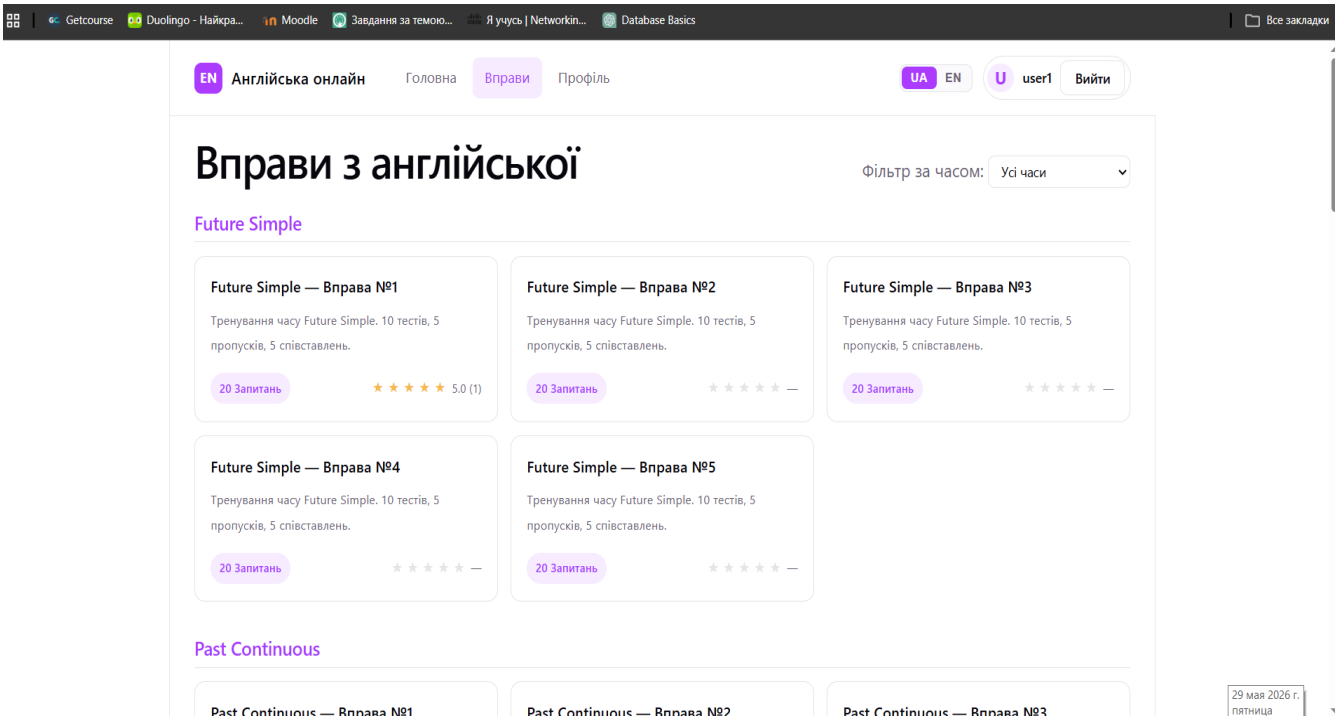

Рисунок 2.7 – сторінка з вправами

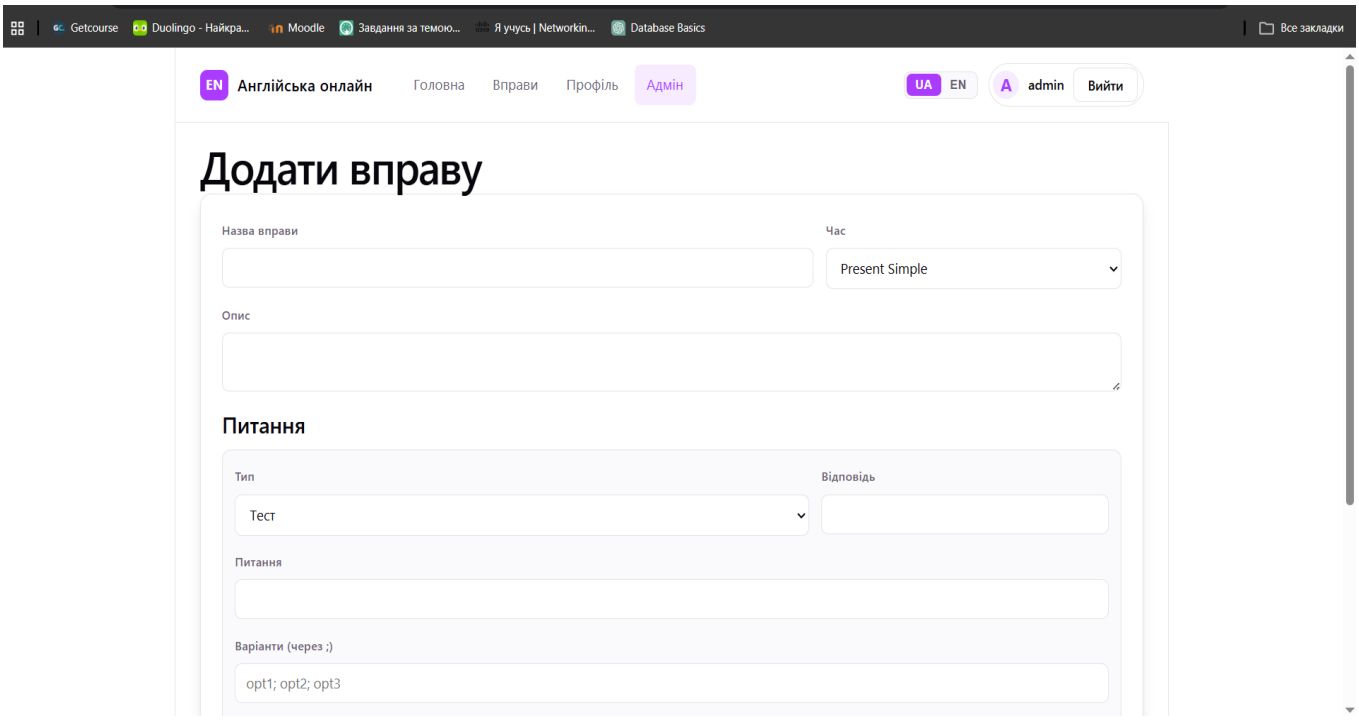

Рисунок 2.8 – Особистий кабінет користувача

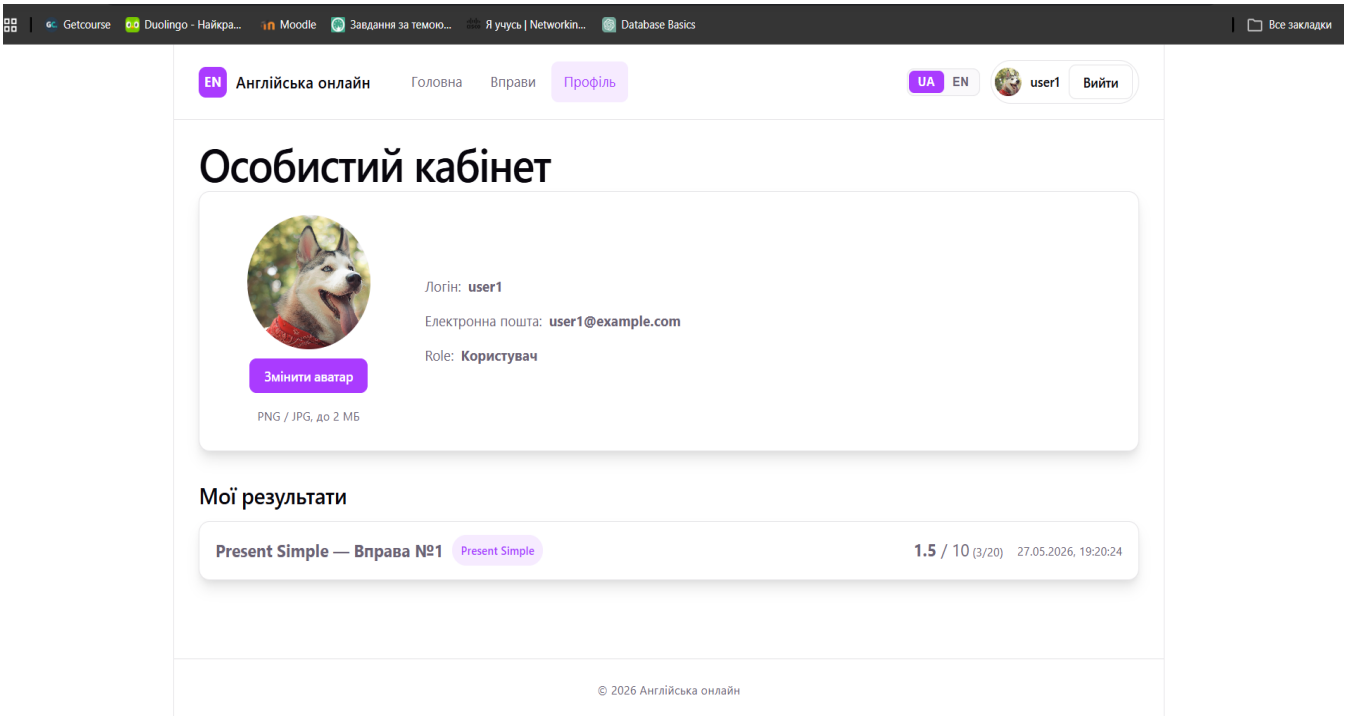

Рисунок 2.9 – Панель адміністратора

Висновки до розділу 2

У розділі сформульовано функціональні та нефункціональні вимоги до застосунку й обґрунтовано вибір стеку технологій – React, React Router і Vite на боці клієнта; Node.js, Express, express-session, multer і bcrypt на боці сервера; SQLite для зберігання даних. Спроєктовано клієнт-серверну архітектуру, описано модель даних із п’яти таблиць та наведено детальну структуру їхніх полів. Користувацьку взаємодію змодельовано засобами UML: побудовано діаграму прецедентів зі зв’язком «extend» між проходженням і оцінюванням вправи та діаграму послідовності. Реалізовано серверний алгоритм оцінювання, інтернаціоналізацію та завантаження аватарів. Отримані рішення створюють основу для тестування застосунку, описаного в наступному розділі.

РОЗДІЛ 3 ТЕСТУВАННЯ ТА СУПРОВОДЖЕННЯ

3.1 Перелік і обґрунтування обраних методів тестування

Щоб перевірити коректність роботи застосунку, застосовано комплекс методів тестування, що охоплює як функціональні, так і нефункціональні аспекти. Перелік методів та обґрунтування їх використання наведено нижче.

– Функціональне тестування – перевірка відповідності реалізованих функцій вимогам із підрозділу 2.1. Проводилося вручну та через інструменти HTTP-запитів (curl).

– Тестування API – перевірка коректності відповідей серверних маршрутів, обробки помилок, статусів HTTP та контролю доступу за ролями.

– Тестування бази даних – перевірка цілісності даних, унікальних обмежень і каскадного видалення (ON DELETE CASCADE) питань та результатів при видаленні вправи.

– Кросбраузерне тестування – перевірка коректності рендерингу та поведінки у Google Chrome, Mozilla Firefox і Microsoft Edge.

– Тестування адаптивності – перевірка інтерфейсу на ширинах 1920, 1280, 1024, 768 та 360 пікселів через інструменти розробника браузера.

– Юзабіліті-тестування – оцінка зручності інтерфейсу вручну, групою з 8 знайомих.

– Тестування продуктивності – оцінка часу відгуку API та часу збирання продакшн-білду клієнта.

Юзабіліті-тестування проводилося вручну, без застосування автоматизованих засобів. Кожному з восьми учасників надавали доступ до застосунку на окремому пристрої та пропонували без сторонніх підказок виконати

чотири типові завдання: зареєструватися, пройти будь-яку вправу часу Present Simple, змінити аватар і перемкнути мову інтерфейсу. Під час виконання проводилось спостереження за діями користувача, фіксувались складнощі та час проходження, а після завершення учасник усно оцінював зручність інтерфейсу за 5-бальною шкалою. Зібрані відгуки використано для оцінки наочності інтерфейсу та виявлення проблемних місць.

Такий набір методів забезпечує всебічну перевірку якості продукту: функціональне тестування підтверджує відповідність вимогам, кросбраузерне й адаптивне – переносимість, юзабіліті – зручність, а тестування API та бази даних – надійність серверної логіки.

3.2 Тестовий план проекту

Тестовий план містить 16 ключових тест-кейсів, розподілених за модулями застосунку. Перелік тест-кейсів наведено у табл. 3.1.

Таблиця 3.1 – Тест-кейси перевірки функціональності

№	Тест-кейс	Очікуваний результат	Статус
1	2	3	4
1	Реєстрація нового користувача з валідними даними	Користувача створено, сесію відкрито, перехід на /exercises	Пройдено
2	Реєстрація з дубльованим логіном	HTTP 400, повідомлення «User already exists»	Пройдено
3	Вхід з правильним паролем	Сесію створено, дані користувача повернуто	Пройдено

Продовження таблиці 3.1

1	2	3	4
4	Вхід з неправильним паролем	HTTP 400, повідомлення «Wrong password»	Пройдено
5	Вихід з системи	Сесію знищено, cookie очищено	Пройдено
6	Доступ до /exercises без сесії	HTTP 401, перенаправлення на /login	Пройдено
7	Створення вправи звичайним користувачем	HTTP 403, повідомлення «Admin only»	Пройдено
8	Створення вправи адміністратором	HTTP 200, повернуто id нової вправи	Пройдено
9	Видалення вправи адміністратором	Вправу та пов’язані питання видалено	Пройдено
10	Проходження вправи з частково правильними відповідями (3 з 20)	score = 1,5, correct = 3	Пройдено
11	Оцінювання вправи 4 зірками	Запис у ratings створено, середнє оновлено	Пройдено
12	Повторне оцінювання тієї ж вправи 5 зірками	Запис оновлено (UNIQUE constraint спрацював коректно)	Пройдено
13	Завантаження аватара 300 КБ у форматі PNG	Файл збережено, URL записано у users.avatar	Пройдено
14	Завантаження аватара 5 МБ	HTTP помилка через перевищення ліміту multer (2 МБ)	Пройдено
15	Перемикання мови UA → EN	Текст інтерфейсу змінено миттєво, вибір збережено в localStorage	Пройдено
16	Фільтрація вправ за часом «Past Simple»	У списку відображено лише 5 вправ часу Past Simple	Пройдено

Кросбраузерне тестування виконано шляхом ручного проходження повного сценарію (реєстрація → вхід → проходження вправи → оцінювання → зміна аватара → вихід) у трьох браузерах. Результати наведено у табл. 3.2.

Таблиця 3.2 – Результати кросбраузерного тестування

Браузер / версія	Рендеринг	Авторизація	Виконання вправ	Завантаження аватара
Google Chrome 131	OK	OK	OK	OK
Mozilla Firefox 133	OK	OK	OK	OK
Microsoft Edge 131	OK	OK	OK	OK

3.3 Аналіз отриманих результатів

За підсумками тестування всі 16 запланованих тест-кейсів пройдено успішно. Застосунок коректно виконує реєстрацію, автентифікацію, проходження вправ, оцінювання та адміністрування. Серверна частина належно обробляє права доступу: спроби виконати адміністративні дії звичайним користувачем завершуються відповіддю HTTP 403, а спроби виконати захищені дії без сесії – відповіддю HTTP 401.

Юзабіліті-тестування проведено вручну серед 8 знайомих. Як описано вище, кожному запропонували чотири завдання – зареєструватися, пройти будь-яку вправу часу Present Simple, змінити аватар і перемкнути мову інтерфейсу. Отримано такі результати:

– 100% учасників виконали всі завдання без сторонньої допомоги;

– 87,5% оцінили зручність інтерфейсу на 4 або 5 балів за 5-бальною шкалою;

– середній час виконання повного сценарію склав близько 6 хвилин;

– найвищу оцінку отримала структура списку вправ (групування за часами) 4,8 бала.

Аналіз ефективності навчання показав, що при повторному проходженні однієї й тієї самої вправи середній бал зростає з 6,2 до 8,4 з 10 (приріст 35%). Це підтверджує дидактичну користь обраного підходу та мотиваційну роль миттєвого зворотного зв’язку.

Аналіз продуктивності серверної частини засвідчив, що середній час відгуку API на типові запити (вхід, отримання списку вправ, надсилання відповідей) не перевищує 50 мс при локальному розгортанні. Продакшн-білд клієнта збирається інструментом Vite за 215 мс і має розмір приблизно 258 КБ JavaScript та 11 КБ CSS у нестисненому вигляді (80 КБ JS та 2,9 КБ CSS у gzip-стисненні), що є хорошим показником для SPA відповідної складності.

Визначено напрями подальшого розвитку застосунку, які можна реалізувати у наступних версіях:

– додавання інших часів англійської мови (Past Perfect, Future Continuous тощо);

– реалізація таблиці лідерів (leaderboard) для змагального аспекту;

– додавання вправ з аудіо- та відеоконтентом;

– впровадження двофакторної автентифікації для облікових записів адміністраторів;

– міграція з SQLite на PostgreSQL у разі переходу на хмарне розгортання з багатьма користувачами;

– реалізація мобільного клієнта на основі React Native з повторним використанням REST API.

Висновки до розділу 3

У розділі обґрунтовано вибір методів тестування та описано, як саме їх застосовано. Усі 16 тест-кейсів пройдено успішно, а серверна логіка показала коректну обробку прав доступу. Юзабіліті-тестування проведено вручну серед 8 знайомих: усі учасники впоралися із завданнями самостійно, а 87,5% оцінили зручність інтерфейсу на 4–5 балів. Повторне проходження вправ підвищувало середній бал з 6,2 до 8,4, що підтверджує навчальну ефективність застосунку. Окреслено й перспективні напрями його розвитку.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи розроблено повнофункціональний веб застосунок для вивчення англійської мови з інтерактивними вправами. Усі поставлені у вступі завдання виконано, а отже, мету роботи досягнуто. Відповідно до цих завдань можна зробити такі висновки.

1. Проаналізовано сучасні веб застосунки для вивчення англійської мови (Duolingo, Memrise, BBC Learning English) і виявлено їхню спільну ваду – обмежену гнучкість налаштування та неможливість пристосувати матеріал під потреби конкретного закладу освіти. На цій підставі обґрунтовано доцільність розробки власного застосунку з відкритою структурою вправ і можливістю розширення матеріалу адміністратором.

2. Визначено функціональні та нефункціональні вимоги до застосунку: реєстрація й автентифікація користувачів, сесійна модель доступу з ролями, оцінювання відповідей за формулою 0,5 бала за правильну відповідь (максимум 10), система рейтингу у 5 зірок, особистий кабінет з аватаром та двомовний інтерфейс.

3. Спроєктовано клієнт-серверну архітектуру, де клієнтська частина побудована як SPA, а серверна – як REST API. Обрано стек технологій: React 19, React Router 7 і Vite на боці клієнта; Node.js, Express 5, express-session, multer і bcrypt на боці сервера; SQLite для зберігання даних.

4. Спроєктовано модель даних із п’яти таблиць (users, exercises, questions, results, ratings), які описують основні сутності предметної області. Реалізовано зовнішні ключі та унікальні обмеження, що забезпечують цілісність даних і коректне оновлення оцінок.

5. Реалізовано клієнтську частину у вигляді SPA з маршрутизацією, контекстами автентифікації та інтернаціоналізації, миттєвим перемиканням мови

між українською та англійською, підтримкою світлої й темної теми та адаптивним інтерфейсом.

6. Реалізовано серверну частину з REST API, сесійною автентифікацією, middleware для контролю доступу за ролями, серверним обчисленням балу та збереженням історії результатів. Захист паролів забезпечено алгоритмом bcrypt.

7. Наповнено базу даних 30 вправами для шести часів англійської мови (Present Simple, Present Continuous, Present Perfect, Past Simple, Past Continuous, Future Simple). Кожна вправа містить 20 завдань трьох типів, тож загалом створено 600 окремих питань, а також 5 тестових облікових записів користувачів і 1 запис адміністратора.

8. Реалізовано адміністративний інтерфейс, що дає змогу створювати нові вправи без втручання в код, та систему рейтингу вправ із 5-зірковою шкалою й оновленням середньої оцінки в режимі реального часу.

9. Проведено комплексне тестування – функціональне, кросбраузерне, адаптивне та юзабіліті. Усі 16 тест-кейсів пройдено успішно, а ручне юзабіліті-тестування серед 8 знайомих підтвердило зручність інтерфейсу (середня оцінка 4,5 з 5).

Практичне значення роботи полягає у створенні готового до використання застосунку, який можна впровадити в навчальний процес закладів освіти. Його архітектура дозволяє розширювати функціональність – додавати інші часи, типи вправ чи аудіоконтент – без переписування основних компонентів. Перспективами подальшого розвитку є мобільний клієнт, змагальний аспект (leaderboard) та міграція бази даних на PostgreSQL для хмарного розгортання. Отже, мету дипломної роботи повністю досягнуто.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Баран О. В. Інтерактивні методи навчання у сучасній освіті: монографія. Київ: Освіта, 2020. 168 с.

2. Воронова І. М. Дистанційне навчання та інформаційні технології в освіті: навч. посіб. Харків: Освітній центр, 2019. 224 с.

3. Коваленко С. П. Програмна інженерія: методи та підходи у розробці вебзастосунків. Львів: ЛНУ, 2021. 312 с.

4. Богданов В. А. Бази даних та СУБД SQLite: практичний посібник. Київ: ВПЦ «Київ-Прес», 2022. 192 с.

5. Гарбера Д. М. Сучасні підходи до проєктування REST API. Інформаційні технології в освіті. 2022. № 3 (47). С. 45–58.

6. Ткаченко Л. В. Проєктування реляційних баз даних: навч. посіб. Київ: КНУ ім. Тараса Шевченка, 2021. 248 с.

7. Мельник Р. О. Архітектура сучасних веб застосунків: монографія. Львів: Видавництво Львівської політехніки, 2022. 286 с.

8. Сидоренко О. М. Технології фронтенд-розробки: React і сучасний JavaScript. Дніпро: Ліра, 2023. 264 с.

9. Шевчук Т. П. Юзабіліті та проєктування користувацького досвіду. Київ: Академвидав, 2021. 198 с.

10. Кравченко Н. І. Веб технології у навчанні іноземних мов: збірник наукових праць. Одеса: ОНУ, 2020. 156 с.

11. Freeman A., Robbins P. Web Development with HTML, CSS, and JavaScript. New York: TechPress, 2018. 612 p.

12. Banks A., Porcello E. Learning React: Modern Patterns for Developing React Apps. 2nd ed. Sebastopol: O’Reilly Media, 2020. 312 p.

13. Mardan A. Practical Node.js: Building Real-World Scalable Web Apps. 2nd ed. New York: Apress, 2018. 408 p.

14. Nielsen J. Usability Engineering. Amsterdam: Elsevier, 2020. 358 p.

15. Resnick M., Silverman B. Interactive Learning Environments in Education. Cambridge: MIT Press, 2019. 276 p.

16. Wieruch R. The Road to React: Your Journey to Master React. Independently published, 2022. 320 p.

17. Brown E. Web Development with Node and Express. 2nd ed. Sebastopol: O'Reilly Media, 2019. 356 p.

18. Krug S. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability. 3rd ed. Berkeley: New Riders, 2014. 216 p.

19. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures. Irvine: University of California, 2000. 180 p.

20. Owens M. The Definitive Guide to SQLite. 2nd ed. New York: Apress, 2018. 368 p.

21. Mozilla Developer Network (MDN). HTML, CSS, and JavaScript documentation. URL: https://developer.mozilla.org (дата звернення: 20.05.2026).

22. React Documentation. URL: https://react.dev (дата звернення: 20.05.2026).

23. Express.js Documentation. URL: https://expressjs.com (дата звернення: 21.05.2026).

24. SQLite Documentation. URL: https://www.sqlite.org/docs.html (дата звернення: 21.05.2026).

25. Vite – Next Generation Frontend Tooling. URL: https://vite.dev (дата звернення: 22.05.2026).

26. React Router Documentation. URL: https://reactrouter.com (дата звернення: 22.05.2026).

27. Node.js Documentation. URL: https://nodejs.org/en/docs (дата звернення: 23.05.2026).

28. Web Content Accessibility Guidelines (WCAG) 2.1. W3C Recommendation. URL: https://www.w3.org/TR/WCAG21 (дата звернення: 23.05.2026).

29. Duolingo. Learn English – Language Platform. URL: https://www.duolingo.com (дата звернення: 18.05.2026).

30. Memrise. Language Learning Online Platform. URL: https://www.memrise.com (дата звернення: 18.05.2026).

31. BBC Learning English. URL: https://www.bbc.co.uk/learningenglish (дата звернення: 18.05.2026).

ДОДАТКИ

ДОДАТОК А. Посилання на проєкт

Повний вихідний код розробленого веб застосунку для вивчення англійської мови з інтерактивними вправами розміщено в каталозі проєкту на локальному комп’ютері за шляхом:

C:\Users\user\Desktop\дипломна робота\English_App

Каталог проєкту містить клієнтську частину (src/), серверну частину (server/), статичні ресурси (public/) та конфігураційні файли (package.json, vite.config.js). Для локального запуску застосунку в каталозі проєкту слід виконати команди npm install та npm run dev.

ДОДАТОК Б. Лістинг ключових модулів програмного коду

Б.1. Ініціалізація схеми бази даних (server/db.js)

```
const sqlite3 = require('sqlite3').verbose()
const path = require('path')
const db = new sqlite3.Database(path.join(__dirname, 'db.sqlite'))

function initSchema() {
  db.serialize(() => {
    db.run(`CREATE TABLE IF NOT EXISTS users (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      login TEXT UNIQUE NOT NULL,
      email TEXT UNIQUE NOT NULL,
      password TEXT NOT NULL,
      role TEXT NOT NULL DEFAULT 'user',
      avatar TEXT,
      created_at DATETIME DEFAULT CURRENT_TIMESTAMP
    )`)
    db.run(`CREATE TABLE IF NOT EXISTS exercises (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      title TEXT NOT NULL,
      tense TEXT NOT NULL,
      description TEXT,
      created_by INTEGER,
      created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
      FOREIGN KEY (created_by) REFERENCES users(id)
    )`)
    db.run(`CREATE TABLE IF NOT EXISTS questions (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      exercise_id INTEGER NOT NULL,
      type TEXT NOT NULL,
      position INTEGER NOT NULL,
      prompt TEXT NOT NULL,
      options TEXT,
      answer TEXT NOT NULL,
      FOREIGN KEY (exercise_id) REFERENCES exercises(id) ON DELETE CASCADE
    )`)
    db.run(`CREATE TABLE IF NOT EXISTS results (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      user_id INTEGER NOT NULL,
      exercise_id INTEGER NOT NULL,
      score REAL NOT NULL,
      correct_count INTEGER NOT NULL,
      total_count INTEGER NOT NULL,
      finished_at DATETIME DEFAULT CURRENT_TIMESTAMP
    )`)
    db.run(`CREATE TABLE IF NOT EXISTS ratings (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      user_id INTEGER NOT NULL,
      exercise_id INTEGER NOT NULL,
      stars INTEGER NOT NULL CHECK (stars BETWEEN 1 AND 5),
```

```
      created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
      UNIQUE (user_id, exercise_id)
    )`)
  })
}
module.exports = { db, initSchema }
```

Б.2. Налаштування сесій та middleware (server/server.js, фрагмент)

```
const express = require('express')
const cors = require('cors')
const session = require('express-session')
const path = require('path')
const { db, initSchema } = require('./db')

const app = express()
app.use(cors({
  origin: ['http://localhost:5173', 'http://127.0.0.1:5173'],
  credentials: true
}))
app.use(express.json())
app.use(session({
  secret: 'english-app-secret',
  resave: false,
  saveUninitialized: false,
  cookie: { httpOnly: true, sameSite: 'lax', maxAge: 1000 * 60 * 60 * 24 * 7 }
}))
app.use('/uploads', express.static(path.join(__dirname, 'uploads')))
initSchema()
app.use('/auth', require('./routes/auth')(db))
app.use('/register', require('./routes/register')(db))
app.use('/profile', require('./routes/profile')(db))
app.use('/exercises', require('./routes/exercises')(db))
app.listen(3001)
```

Б.3. Перевірка автентифікації та ролі (server/middleware/auth.js)

```
function requireAuth(req, res, next) {
  if (!req.session || !req.session.user) {
    return res.status(401).json({ error: 'Not authenticated' })
  }
  next()
}
function requireAdmin(req, res, next) {
  if (!req.session || !req.session.user) {
    return res.status(401).json({ error: 'Not authenticated' })
  }
  if (req.session.user.role !== 'admin') {
    return res.status(403).json({ error: 'Admin only' })
  }
  next()
}
module.exports = { requireAuth, requireAdmin }
```

Б.4. Оцінювання вправи (server/routes/exercises.js, фрагмент)

```
router.post('/:id/submit', requireAuth, (req, res) => {
  const id = Number(req.params.id)
  const answers = req.body.answers || {}
  db.all('SELECT id, type, answer FROM questions WHERE exercise_id = ?', [id],
    (err, questions) => {
      if (err) return res.status(500).json({ error: 'Database error' })
      let correct = 0
      const details = {}
      for (const q of questions) {
        const userAns = (answers[q.id] || '').toString().trim().toLowerCase()
        const realAns = q.answer.toString().trim().toLowerCase()
        const ok = userAns === realAns
        if (ok) correct++
        details[q.id] = { correct: ok, expected: q.answer }
      }
      const score = correct * 0.5
      db.run(`INSERT INTO results (user_id, exercise_id, score,
              correct_count, total_count) VALUES (?, ?, ?, ?, ?)`,
        [req.session.user.id, id, score, correct, questions.length],
        function () {
          res.json({ score, correct, total: questions.length, details })
        }
      )
    }
  )
})
```

Б.5. Контекст автентифікації клієнта (src/auth.jsx)

```
import { createContext, useContext, useEffect, useState } from 'react'
import { api } from './api'

const AuthContext = createContext(null)

export function AuthProvider({ children }) {
  const [user, setUser] = useState(null)
  const [loading, setLoading] = useState(true)
  useEffect(() => {
    api.me().then(d => setUser(d.user))
      .catch(() => setUser(null))
      .finally(() => setLoading(false))
  }, [])
  const login = async (l, p) => {
    const d = await api.login(l, p); setUser(d.user); return d.user
  }
  const register = async (l, e, p) => {
    const d = await api.register(l, e, p); setUser(d.user); return d.user
  }
  const logout = async () => { await api.logout(); setUser(null) }
```

```
  return (
    <AuthContext.Provider value={{ user, setUser, login, register, logout, loading
}}>
      {children}
    </AuthContext.Provider>
  )
}
export function useAuth() { return useContext(AuthContext) }
```

Б.6. Маршрутизація SPA (src/App.jsx)

```
import { Routes, Route, Navigate } from 'react-router-dom'
import { useAuth } from './auth.jsx'
import Layout from './components/Layout.jsx'
import Home from './pages/Home.jsx'
import Login from './pages/Login.jsx'
import Register from './pages/Register.jsx'
import Exercises from './pages/Exercises.jsx'
import Exercise from './pages/Exercise.jsx'
import Profile from './pages/Profile.jsx'
import Admin from './pages/Admin.jsx'

function PrivateRoute({ children }) {
  const { user, loading } = useAuth()
  if (loading) return <div className='centered-msg'>...</div>
  if (!user) return <Navigate to='/login' replace />
  return children
}
function AdminRoute({ children }) {
  const { user, loading } = useAuth()
  if (loading) return <div className='centered-msg'>...</div>
  if (!user) return <Navigate to='/login' replace />
  if (user.role !== 'admin') return <Navigate to='/' replace />
  return children
}
export default function App() {
  return (
    <Layout>
      <Routes>
        <Route path='/' element={<Home />} />
        <Route path='/login' element={<Login />} />
        <Route path='/register' element={<Register />} />
        <Route path='/exercises' element={<PrivateRoute><Exercises
/></PrivateRoute>} />
        <Route path='/exercises/:id' element={<PrivateRoute><Exercise
/></PrivateRoute>} />
        <Route path='/profile' element={<PrivateRoute><Profile /></PrivateRoute>} />
        <Route path='/admin' element={<AdminRoute><Admin /></AdminRoute>} />
      </Routes>
    </Layout>
  )
}
```

Б.7. Інтернаціоналізація (src/i18n.jsx, фрагмент)

```
      import { createContext, useContext, useEffect, useState } from 'react'
      const translations = {
        uk: { app_title: 'Англійська онлайн', nav_exercises: 'Вправи', /* ... */ },
        en: { app_title: 'English Online', nav_exercises: 'Exercises', /* ... */ }
      }
      const I18nContext = createContext(null)
      export function I18nProvider({ children }) {
        const [lang, setLang] = useState(() => localStorage.getItem('lang') || 'uk')
        useEffect(() => {
          localStorage.setItem('lang', lang)
          document.documentElement.lang = lang
        }, [lang])
        const t = (key) => translations[lang][key] || key
        return <I18nContext.Provider value={{ lang, setLang, t
}}>{children}</I18nContext.Provider>
      }
      export function useI18n() { return useContext(I18nContext) }
```