Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ВЕБДОДАТКУ ДЛЯ ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ З ВИКОРИСТАННЯМ REACT, TAILWINDCSS ТА MYSQL

DEVELOPMENT OF A WEB APPLICATION FOR LEARNING ENGLISH USING REACT, TAILWINDCSS AND MYSQL

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗ-41
Малевич Юрій Валерійович

Керівник:
Ліщина Наталія Миколаївна

Кваліфікаційну роботу
допущено до захисту
«10» 06 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна
_____

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«20» 12 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Малевичу Юрію Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *«Розробка вебдодатку для вивчення англійської мови з використанням React, TailwindCSS та MySQL»*
Керівник роботи: *к.т.н., доцент Ліщина Н. М.*
затверджені наказом закладу вищої освіти від *«19» грудня 2024 р.* № *474/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра *«10» червня 2025 р.*
3. Вихідні дані до роботи: *Visual Studio Code, React.js, TailwindCSS, SCSS, HTML5, JavaScript (ES6+), Node.js, Express.js, MySQL, ORM Sequelize, Git, Postman, методичні вказівки до виконання кваліфікаційної роботи бакалавра.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *проаналізувати сучасні рішення, сформулювати вимоги до продукту, обґрунтувати вибір технологій, розробити архітектуру, розробити ключові модулі, впровадити систему авторизації, провести тестування, реалізувати стабільне відображення на хостингу.*
5. Перелік графічного матеріалу: *14 рисунків, 1 додаток*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Ліщина Н. М.*		
Специфікація вимог до розробленої системи	*Ліщина Н. М.*		
Розробка об'єкта проектування	*Ліщина Н. М.*		
Нормоконтроль	*Вознюк А. В.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	1,57 %		
Академічна доброчесність	*Ліщина Н. М.*		

7. Дата видачі завдання *«20» грудня 2024 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 04.02.2025 р.	викон.
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 01.03.2025 р.	викон.
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 15.03.2025 р.	викон.
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	до 29.03.2025 р.	викон.
5	Практична реалізація об'єкта проектування та розробка бази даних	до 26.04.2025 р.	викон.
6	Тестування та налагодження об'єкта проектування	до 03.05.2025 р.	викон.
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	до 10.06.2025 р.	викон.

Здобувач вищої освіти Юрій МАЛЕВИЧ

Керівник кваліфікаційної роботи Наталія ЛІЩИНА

АНОТАЦІЯ

Малевич Ю. В. Розробка вебдодатку для вивчення англійської мови з використанням React, TailwindCSS та MySQL. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

У першому розділі проаналізовано предметну область, виявлено недоліки існуючих рішень та сформульовані завдання на кваліфікаційну роботу.

У другому розділі визначено вимоги до функціональності, архітектури та інтерфейсу вебдодатку, проведено вибір засобів розробки і побудовано модель інформаційної системи.

Третій розділ присвячено реалізації інтерфейсу, серверної логіки, бази даних, а також тестуванню, налагодженню, захисту системи та аналізу гнучкості та масштабованості системи.

У висновках підсумовано отримані результати, обґрунтовано доцільність застосованих рішень і визначено перспективи подальшого розвитку та впровадження системи в освітнє середовище.

Ключові слова: вебдодаток, кваліфікаційна робота, вивчення англійської мови, React, TailwindCSS, MySQL, база даних, тестування, адаптивність.

ABSTRACT

Malevych Y. Development of a web application for learning English using React, TailwindCSS and MySQL. Manuscript. Qualification work of the bachelor's degree program "Software Engineering", specialty "Software Engineering". Lutsk National Technical University. Lutsk, 2025.

The bachelor's thesis consists of an introduction, three chapters, conclusions, and a list of references and applications.

The first section analyzes the subject area, identifies the shortcomings of existing solutions and formulates the tasks for the qualification work.

The second section defines the requirements for the functionality, architecture and interface of the web application, selects development tools and builds a model of the information system.

The third section is devoted to the implementation of the interface, server logic, database, as well as testing, debugging, system security, and analysis of the system's flexibility and scalability.

The conclusions summarize the results obtained, substantiate the feasibility of the applied solutions, and determine the prospects for further development and implementation of the system in the educational environment.

Keywords: web application, qualification work, learning English, React, TailwindCSS, MySQL, database, testing, adaptability.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі володіння англійською мовою є важливою умовою академічного та професійного зростання. Особливої актуальності набувають інформаційно-комп’ютерні системи, що дають змогу самостійно здобувати знання в різних форматах. Попри велику кількість вже існуючих вебплатформ для вивчення мови, більшість із них мають суттєві обмеження: неповну функціональність у безкоштовному доступі, відсутність персоналізації, недостатню підтримку українськомовної аудиторії, неможливість адаптації під індивідуальні потреби кожного користувача.

Світові тенденції вирішення зазначених проблем включають розробку адаптивних, масштабованих та персоналізованих освітніх платформ із використанням сучасних фреймворків, утилітарних бібліотек, а також реляційних баз даних. Актуальність розробки полягає у створенні інструменту, що поєднує повноцінне навчання, збереження результатів, тестування та сертифікацію в одному рішенні, орієнтованому на україномовну аудиторію.

Метою кваліфікаційної роботи є розробка функціонального, адаптивного та безпечного вебдодатку для вивчення англійської мови, побудованого на основі стеку, що дозволяє забезпечити комфортне та ефективне засвоєння навчального матеріалу, відстеження прогресу користувачів та інтеграцію з системами збереження і обробки даних.

Об’єктом кваліфікаційної роботи є процес організації дистанційного навчання іноземним мовам за допомогою інформаційних систем.

Предметом кваліфікаційної роботи виступають методи, технології та інструменти створення вебдодатку для вивчення англійської мови, що реалізує повний навчальний цикл – від теоретичного ознайомлення до контролю і сертифікації.

У межах виконання роботи передбачається досягнення таких конкретних цілей:

– проаналізувати сучасні вебплатформи для вивчення англійської мови, виявити їх переваги та недоліки;

– сформулювати вимоги до функціональності, дизайну та бази даних майбутнього додатку;

– обґрунтувати вибір технологічного стеку для реалізації клієнтської, серверної та інформаційної частин системи;

– розробити архітектуру вебдодатку з урахуванням принципів масштабованості, розширюваності та безпеки;

– реалізувати ключові модулі: словник, граматика, аудіювання, тестування, сертифікація;

– впровадити систему реєстрації та авторизації користувачів;

– провести функціональне тестування вебдодатку, перевірити його стабільність, адаптивність та відповідність поставленим вимогам;

– реалізувати стабільне відображення на хостингу.

З метою апробації матеріалів роботи підготовлено тези, які були подані до збірнику тез «Міжнародної науково-практичної конференції ІТОНВ-2025» (м. Луцьк, 23-24 травня 2025 року) (додаток А).

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Вивчення англійської мови в останні роки зазнало значних змін під впливом інформаційних технологій. Зростання кількості користувачів цифрових пристроїв, поширення доступу до високошвидкісного інтернету та необхідність здобувати нові знання без прив’язки до навчальних закладів стимулювали попит на вебзастосунки, які дозволяють ефективно, зручно і швидко опановувати іноземні мови. Великого значення це набуває в українському контексті, де потреба у мовній підготовці суттєво зросла на тлі глобальних процесів, академічної мобільності та інтеграції у міжнародне освітнє середовище.

Англійська мова є провідною мовою науки, технологій, бізнесу, IT-індустрії, тому її знання стало необхідною умовою не лише для професійного розвитку, а й для повноцінної участі у житті цифрового суспільства нашого часу. Класичні моделі вивчення іноземної мови у вигляді занять з викладачем або на мовних курсах дедалі більше поступаються місцем комбінованим або повністю дистанційним форматам навчання, що передбачають взаємодію з навчальними системами, адаптованими під індивідуальний темп, рівень знань, мету вивчення та доступний час користувача. Саме тому особливого значення набувають вебдодатки, які дозволяють інтегрувати в одне середовище все, чого потребує користувач.

На ринку цифрових освітніх послуг сформувались списки платформ, що пропонують рішення для самостійного вивчення англійської мови. Найбільш відомими серед них є Duolingo, Babbel, BBC Learning English. Ці сервіси демонструють різні підходи до організації навчання користувача. Duolingo, наприклад, базується на гейміфікованій подачі матеріалу з використанням рівнів, балів і нагород. Babbel – на структуроване вивчення граматики та лексики. BBC Learning English забезпечує доступ до записаних аудіо та відеоматеріалів з

автентичним контентом. Кожна з цих систем має власні переваги, однак аналіз їх функціональності (рис. 1.1) виявляє низку спільних проблем, які знижують ефективність навчального процесу в умовах реального використання.

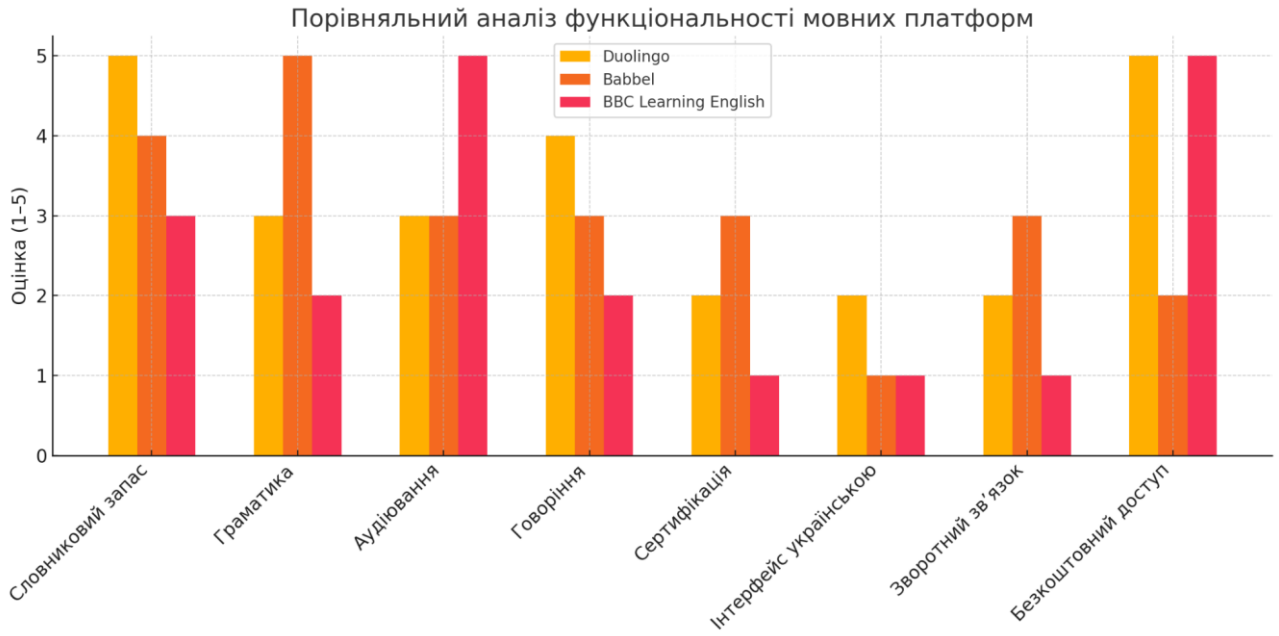

Рисунок 1.1 – Порівняльна оцінка функціональності платформ Duolingo, Babbel, BBC Learning English

Більшість популярних платформ мають обмежений безкоштовний функціонал. Користувачі змушені оформлювати платну підписку, щоб отримати доступ до повного обсягу навчальних матеріалів, історії прогресу, тестування або сертифікатів. Це обмежує доступ до якісного навчального контенту для тих, хто не має фінансової можливості оплачувати підписку, що є особливо актуальним для студентів.

Існуючі системи часто зосереджені лише на окремих частинах мовної компетентності. Одні платформи роблять акцент виключно на словниковому запасі, інші – на граматичних правилах, навичках читання чи прослуховування. Комплексного рішення, що поєднує різні типи мовленнєвої діяльності та забезпечує всебічний розвиток навичок, бракує. Крім того, у багатьох додатках відсутній логічний зв’язок між модулями і інтерфейсом: користувач не має змоги

системно проходити навчання за доступною програмою, що поступово ускладнюється та закріплюється. Цю проблему зможе вирішити React [1].

Ще однією суттєвою проблемою є відсутність повноцінного зворотного зв’язку. Навчання без отримання пояснень, підказок або коментарів до помилок, ускладнює засвоєння матеріалу, особливо для початківців. Крім того, велика частина платформ не підтримує збереження детальної статистики користувача, яка б дозволила аналізувати власний прогрес, виявляти слабкі місця та повторювати складні теми. Вони мають бути адаптивними для різних пристроїв, що покращить доступність – це буде досягнуто за допомогою SCSS [2]. Без цього втрачається можливість персоналізації навчального процесу, що є однією з ключових переваг цифрових рішень.

Варто зазначити низьку доступність існуючих рішень для україномовної аудиторії. Більшість популярних платформ не мають україномовного інтерфейсу, що створює додатковий бар’єр для початківців, які не володіють англійською на рівні, достатньому для самостійного розуміння інструкцій, завдань, зображень або пояснень. Відсутність локалізації ускладнює використання таких сервісів у школах, коледжах або університетах з українською мовою викладання. Це знижує ефективність платформ у контексті формальної освіти та унеможливлює їх інтеграцію в вже доступні навчальні програми.

Потрібно згадати про технічну обмеженість більшості наявних платформ. Їхня архітектура зазвичай закрита: користувач не має змоги змінювати структуру курсу, додавати власні коментарі, змінювати складність завдань або впроваджувати додаткові функції моніторингу прогресу. Це виключає можливість використання платформи як гнучкого середовища для викладачів, які хочуть створити власні навчальні маршрути. Більше того, такі системи часто не передбачають повноцінної авторизації з профілем користувача, що зберігає його дані у хмарі, а також не підтримують генерацію електронних сертифікатів про проходження курсу, що є важливим показником.

На фоні описаних недоліків простежуються глобальні тенденції, які визначають напрямок розвитку освітніх вебтехнологій. Серед них – орієнтація на відкриту архітектуру, яка дозволяє налаштовувати систему під різні потреби, використання компонентних підходів до побудови інтерфейсу, впровадження адаптивного дизайну для всіх типів використовуємих пристроїв, інтеграція з базами даних для збереження інформації, реалізація функціоналу персонального кабінету та сертифікації, захист персональних даних і використання інструментів для тестування та валідації на кожному етапі взаємодії з користувачем.

У межах кваліфікаційної роботи передбачено реалізацію вебдодатку, що вирішує вищевказані проблеми та впроваджує найкращі рішення сучасної веброзробки. Система має охоплювати повноцінний цикл навчання: від ознайомлення з теоретичним матеріалом до інтерактивних завдань і тестування. Вебдодаток буде мати україномовний інтерфейс, зручну реєстрацію та збереження прогресу, можливість отримати сертифікат, а також буде підтримувати масштабування, розширення функціональності та адаптацію під інші можливі предметні області.

Таким чином, аналіз сучасного стану наявних цифрових рішень у сфері вивчення англійської мови свідчить про суттєві прогалини, які потребують вирішення через створення нового, адаптивного, повнофункціонального та доступного вебдодатку. Такий застосунок має стати дієвим інструментом для користувачів, що прагнуть вивчати англійську мову самостійно, ефективно та з можливістю контролю власного прогресу.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Метою кваліфікаційної роботи є розробка функціонального та адаптивного вебдодатку для вивчення англійської мови, який поєднує зрозумілий інтерфейс, структурований навчальний контент, якісну систему контролю знань і збереження результатів користувача. Такий ресурс повинен бути орієнтований

на україномовну аудиторію та відповідати сучасним вимогам до вебсистем у сфері онлайн-освіти і розробки. ких конкретних цілей:

– проаналізувати сучасні вебплатформи для вивчення англійської мови, виявити їх переваги та недоліки;

– сформулювати вимоги до функціональності, дизайну та бази даних майбутнього додатку;

– обґрунтувати вибір технологічного стеку для реалізації клієнтської, серверної та інформаційної частин системи;

– розробити архітектуру вебдодатку з урахуванням принципів масштабованості, розширюваності та безпеки;

– реалізувати ключові модулі: словник, граматика, аудіювання, тестування, сертифікація;

– впровадити систему реєстрації та авторизації користувачів;

– провести функціональне тестування вебдодатку, перевірити його стабільність, адаптивність та відповідність поставленим вимогам;

– реалізувати стабільне відображення на хостингу.

Досягнення цілей дозволить створити повноцінну освітню платформу, що відповідає актуальним вимогам до зручності інтерфейсу, функціональності, безпеки та ефективності навчального процесу. Результати роботи продемонструють практичне застосування сучасних вебтехнологій у вирішенні завдання цифрової трансформації мовної освіти.

Висновки до розділу 1

У результаті проведеного аналізу предметної області встановлено, що на сучасному ринку існує велика кількість вебплатформ для вивчення англійської мови, проте більшість з них мають обмеження щодо функціональності, доступності або адаптованості до потреб користувачів. Проведено порівняння схожих систем, виявлено їх сильні та слабкі сторони, що дозволило визначити доцільність створення нового рішення.

Сформульовано мету та поставлено перелік конкретних задач, які мають бути вирішені у межах кваліфікаційної роботи. Було запропоновано напрям розробки майбутнього вебдодатку, що поєднуватиме інтерактивні освітні можливості, зручний інтерфейс та сучасний технологічний стек. Результати цього розділу стали основою для подальшого етапу створення архітектури, вибору інструментів реалізації та формування структури програмної системи.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

На етапі проєктування будь-якої інформаційної системи важливо не лише чітко сформулювати завдання, які вона має вирішувати, але й детально визначити вимоги до її функціональності, структури, архітектури, інтерфейсу та безпеки. При створенні вебдодатку для вивчення англійської мови ці аспекти набувають особливого значення, адже система орієнтована на широку аудиторію, яка очікує швидкого, інтуїтивно зрозумілого інтерфейсу та результативного навчального процесу. Якість формулювання вимог на цьому етапі визначає не лише успішність розробки, а й зручність подальшого використання програмного продукту, його стабільність і масштабованість.

Для забезпечення коректної реалізації функцій навчального процесу у вебдодатку, перед початком технічної реалізації було проведено аналіз предметної області, у межах якого вивчено вже наявні рішення, досвід користування подібними платформами. Це дозволило сфокусуватися на дійсно актуальних потребах і вилучити надлишкову функціональність, що могла б лише ускладнити взаємодію з системою.

Як модель проєкту було обрано модифікований каскадний підхід, який, на відміну від класичної «водоспадної» моделі, передбачає можливість повернення до попередніх етапів у разі виникнення змін або уточнень. Такий підхід є виправданим у випадках, коли програмна система створюється з нуля і включає ряд незалежних, але взаємопов’язаних модулів, як у цій роботі. Модульність системи дозволяє масштабувати її без потреби в суттєвих змінах на всіх рівнях і полегшує майбутнє масштабування.

Під час збору вимог до майбутнього програмного продукту були використані такі методи: аналіз функціоналу популярних освітніх платформ, вивчення коментарів і відгуків користувачів в онлайн-спільнотах, а також

проведення бесід з викладачами англійської мови. Усі ці методи сприяли формуванню актуального та практично орієнтованого набору вимог, що максимально відповідає очікуванням цільової аудиторії схожих застосунків.

Цілі розробки включають створення сучасного інтерактивного середовища для самостійного вивчення англійської мови, яке дозволить користувачам поступово проходити навчальний матеріал, виконувати вправи, перевіряти рівень засвоєних знань, а також зберігати власні результати в особистому кабінеті. Навчальний контент структуровано за ключовими напрямами – лексика, граматика, аудіювання, тестування. Це дозволяє користувачу концентруватися на певному типі навичок або комбінувати їх у межах одного курсу. Крім того, система має реалізовувати створення сертифікатів для тих, хто успішно пройшов усі модулі.

Функціональні вимоги до системи охоплюють наступні можливості:

– реєстрація нового користувача з введенням даних, а також авторизація вже раніше зареєстрованого користувача з наданням доступу до персонального кабінету;

– перегляд курсу, вибір теми та її проходження у форматі інтерактивних блоків;

– проходження тестів, які автоматично оцінюються системою та заносяться у профіль користувача;

– відображення статистики успішності, поточних результатів та рівня прогресу у вивченні курсу;

– отримання сертифіката, що підтверджує завершення курсу, з можливістю його завантаження у форматі PDF.

Нефункціональні вимоги є не менш важливими, адже саме вони забезпечують комфорт, стабільність і доступність у щоденному використанні системи. Зокрема, вебдодаток має бути повністю адаптивним, з коректним відображенням інтерфейсу на всіх видах пристроїв.

Особливу увагу приділено швидкості обробки запитів та зменшенню затримок від серверу при взаємодії з клієнтською частиною. Проведена робота з

аналізу вимог та попереднього моделювання дозволяє впевнено перейти до етапу детального технічного проєктування системи, вибору відповідного інструменту та реалізації логіки.

З архітектурне рішення проєкту буде організовано за класичною трирівневою моделлю: фронтенд – сервер – база даних. Кожен рівень функціонує незалежно, обмінюючись даними через HTTP-запити. Такий підхід дозволяє чітко розподілити відповідальність між обробкою інтерфейсу, логікою та зберіганням даних, що суттєво полегшує масштабування системи, її супровід і тестування кожного модуля. Крім того, трирівнева архітектура надає можливість незалежного оновлення кожного з компонентів без впливу на роботу інших частин застосунку. Для ілюстрації цієї архітектури створено діаграми:

– класів – для відображення сутностей, таких як користувач і його токен доступу (рис. 2.1);

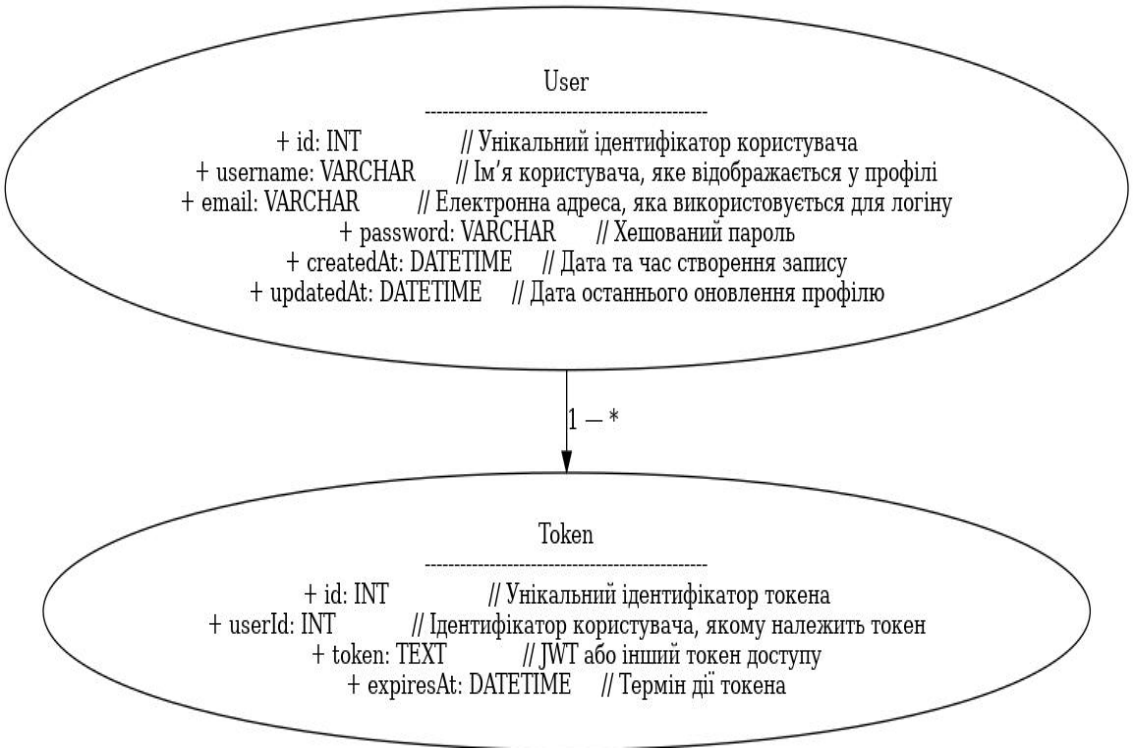

Рисунок 2.1 – Зображення класу сутностей

– алгоритмічні блок-схеми – для опису логіки перевірки авторизації або реєстрації (рис. 2.2).

Рисунок 2.2 – Блок-схема алгоритму авторизації

На основі цього пункту формуються передумови для ефективного виконання подальших етапів розробки.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Проєктування та розробка програмного забезпечення вимагає чіткого вибору інструментів, які забезпечують не лише відповідність функціональним вимогам, а й зручність подальшої підтримки, масштабування та ефективності розгортання. З огляду на поставлену задачу – створення сучасного вебдодатку для вивчення англійської мови – було проведено порівняльний аналіз наявних інструментів, бібліотек, мов програмування, а також методів побудови

архітектури й управління даними. Обрані рішення повинні забезпечити надійність, продуктивність, безпеку і високий рівень користувацького досвіду.

У якості фронтенд-фреймворку обрано React [1] – популярну JavaScript-бібліотеку, призначену для створення інтерфейсів користувача. Серед її головних переваг – компонентно-орієнтована архітектура, яка дозволяє будувати UI як набір незалежних модулів з можливістю повторного використання. React також підтримує декларативний підхід до опису інтерфейсу, а використання віртуального DOM пришвидшує оновлення сторінки та мінімізує взаємодію з реальним DOM.

Для опису стилів в основі використовуватиметься SCSS (Sassy CSS) [2] – препроцесор для CSS, який дозволяє створювати гнучкі стилі з можливістю підтримки за рахунок змінних, вкладеності, міксинів та логіки у вигляді умовних конструкцій. Був проведений аналіз нативного CSS і SCSS (рис. 2.3). Використання SCSS дасть змогу краще структурувати стилі згідно з принципами BEM (Block-Element-Modifier), що позитивно позначиться на подальшій підтримці та читабельності коду.

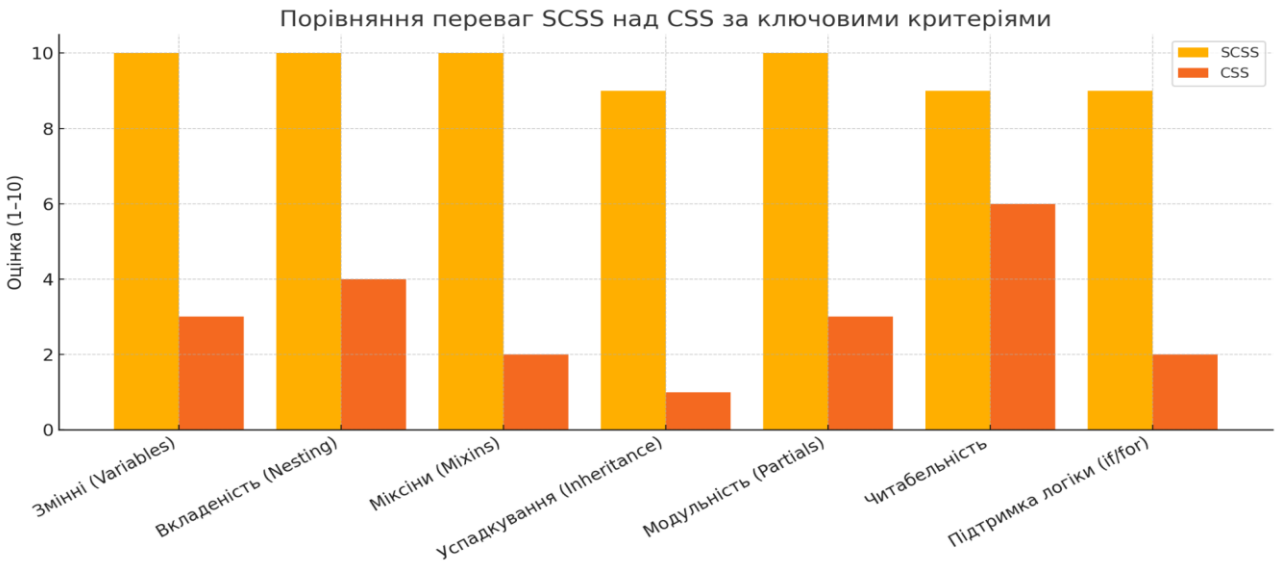

Рисунок 2.3 – Порівняння функціональних можливостей SCSS і CSS за ключовими критеріями

У системі також буде присутнім TailwindCSS [3] – не як основний фреймворк для стилізації, а як засіб оптимізації фінального документа стилю CSS. Зокрема, за допомогою механізму purge (tree-shaking) з фінального CSS-файлу видаляються всі неактивні стилі, що дозволяє значно зменшити обсяг коду, який передається на клієнт. Це позитивно вплине на швидкість завантаження сторінок, загальну продуктивність додатку та підтримку на слабких пристроях.

Для реалізації серверної логіки обрано Node.js [4] – JavaScript-платформу, яка дозволяє виконувати код на серверній стороні. Це дасть змогу зберегти стандарт мови як на клієнті, так і на сервері. Основою серверного застосунку стане Express.js [5] – мінімалістичний фреймворк, що дозволяє використанням middleware-архітектури. Через Express реалізується маршрутизація запитів, обробка форм, надсилання листів, зв’язок з базою даних, обробка помилок та інші моменти серверної логіки.

Для збереження облікових записів користувачів та авторизації планується використовувати базу даних MySQL [6]. У межах створення серверної логіки може виникнути потреба у стабільному та хмарному середовищі для зберігання користувацьких даних. Для запобігання було обрано Aiven for MySQL [7] – повноцінну керовану реляційну базу даних у хмарі, яка надає надійний, масштабований та безпечний сервіс з підтримкою стандартного SQL. Це дозволить уникнути ручного адміністрування сервера бази даних, налаштувань бекапів, моніторингу та безперервної підтримки. Aiven буде використовуватися для реалізації реєстрації та авторизації користувачів. Зокрема, зберігатиметься логін, пароль, дата створення облікового запису та базовий профіль користувача. База буде організована у вигляді окремої таблиці, яка містить унікальний ідентифікатор, e-mail, пароль та інші службові поля. Такий підхід дозволить створити контроль доступу до системи та забезпечити стійке зберігання даних навіть у випадку перезапуску серверної частини.

Середовище розробки проєкту обране з урахуванням продуктивності, зручності та сучасних практик. Основним інструментом розробки стане Visual

Studio Code [8] – легка, гнучка й розширювана IDE, яка дозволить ефективно працювати з фронтендом, бекендом і конфігураційними файлами. Використовуватиметься розширення ESLint для контролю якості коду, Prettier для форматування, Live Server для перевірки змін у реальному часі. Для спрощення роботи з великим обсягом стилів, які будуть написані з використанням SCSS, у середовищі розробки Visual Studio Code додатково застосовуватиметься плагін eCSStractor.

Для контролю версій зручно використовувати Git [9], що дасть змогу фіксувати ключові етапи розробки, повертатися до стабільних версій, а також ізолювати роботу над окремими модулями у гілках без впливу на основну версію проєкту. Розміщення репозиторію на GitHub забезпечить надійне зберігання коду в хмарі, а також дозволить у майбутньому підключити CI/CD-інструменти для автоматизації тестування. Коміти будуть структуровані за змістом, з описами змін і функціоналу, що дозволить легко орієнтуватися в історії змін.

Для тестування серверних запитів використовуватиметься Postman [10] – зручний інструмент, який дозволяє надсилати HTTP-запити до API, перевіряти відповіді, статус-коди і обробку помилок. З його допомогою буде перевірена робота реєстрації та авторизації користувача, а також взаємодія з базою даних у реальному режимі.

Під час розробки, налагодження та подальшого тестування функціональності вебдодатку особливу увагу буде приділено перевірці його роботи у найбільш популярних сучасних браузерах – Google Chrome та Mozilla Firefox. Обидва браузери є кросплатформенним рішенням, які є повністю безкоштовними. В них підтримуються сучасні стандарти та наявні потужні вбудовані інструменти розробника, які дозволяють виконувати повний аналіз поведінки клієнтської частини вебзастосунку.

Google Chrome є браузером з найбільшою часткою користувачів у світі, що робить його основною платформою для тестування сумісності, швидкодії та взаємодії з інтерфейсом користувача. Його DevTools включає в себе модулі для перегляду DOM, стилів CSS, мережевих запитів, продуктивності сторінки,

JavaScript-консолі, емуляції мобільних пристроїв тощо. Це дозволяє у режимі реального часу відслідковувати зміни в інтерфейсі, перевіряти застосування класів, стилів, обробку подій, відповідь серверу, а також симулювати і аналізувати навантаження на систему при взаємодії з елементами UI.

Mozilla Firefox також широко використовується як серед звичайних користувачів, так і серед розробників, зокрема завдяки своїй орієнтації на безпеку, відкритий код та стабільну підтримку нових технологій. Інструменти Firefox Developer Tools включають аналогічний набір функцій для роботи з HTML, CSS, JavaScript, HTTP-запитами та зберіганням локальних даних (LocalStorage, SessionStorage, Cookies). Firefox також дозволяє проводити аналіз продуктивності та пам’яті, здійснювати виклики функцій, переглядати медіа-запити, а також покроково налагоджувати JavaScript-код за допомогою debugger.

Обидва браузери підтримують режим емуляції пристроїв із різними розмірами екранів, що дозволяє перевірити адаптивність верстки згідно принципів UI/UX. У процесі тестування буде досліджено коректність відображення компонентів інтерфейсу на різних ширинах вікна, перевірено відповідність до медіа-запитів CSS, адаптацію шрифтів, масштабування зображень, поведінку елементів навігації, форм, кнопок, сіткової структури тощо.

Також у рамках перевірки буде проводитися моніторинг консолі браузера на наявність помилок у JavaScript. Всі некоректні або заблоковані запити, порушення політики безпеки контенту та інші потенційні проблеми, які можуть вплинути на коректну роботу додатку – всю цю інформацію можна переглянути. Буде протестовано відповідь системи на взаємодії користувача – кліки, введення даних, надсилання форм, навігацію сторінками, роботу модальних вікон, теплові карти. Це дозволить переконатися у правильності роботи подій, відповідності логіки, перевірці валідації введених даних, а також ефективності маршрутизації та динамічного оновлення контенту.

Окремо буде звернено увагу на продуктивність клієнтської частини: час завантаження сторінки, кількість мережевих запитів, обсяг можливого трафіку,

розмір завантажених скриптів та стилів, оптимізацію зображень і ресурсів. За допомогою вкладок Performance та Lighthouse в Chrome або аналогічних інструментів у Firefox буде зібрано дані про час рендерингу, затримки взаємодії та рекомендації щодо оптимізації продуктивності в цілому.

Для хостингу серверної частини розробленого вебдодатку було обрано платформу Render [11] – сучасне рішення класу PaaS (Platform as a Service), що забезпечує швидке, надійне й автоматизоване розгортання серверної інфраструктури на хостингу. Render дозволяє створювати Web Services, які працюють у постійно активному середовищі з підтримкою HTTP/HTTPS, автоматичного масштабування та моніторингу. Завдяки вбудованій інтеграції з системою керування версіями Git, зокрема з GitHub, розгортання додатку відбувається безперервно – кожне оновлення головної гілки репозиторію автоматично ініціює процес нового білду та нової публікації. Це значно спрощує підтримку додатку в актуальному стані, мінімізуючи людський фактор та потребу в постійних ручних налаштуваннях.

Render підтримує середовище виконання Node.js, що дозволяє безпосередньо запускати Express-сервер, необхідний для обробки API-запитів, керування базою даних та взаємодії з клієнтською частиною. У рамках автоматичного розгортання Render виконує встановлення залежностей та запуск основного файлу застосунку, базуючись на конфігурації, визначеній у package.json або в налаштуваннях сервісу. Платформа також надає зручну панель для перегляду логів, стану сервера, метрик споживання ресурсів, що дозволяє оперативно виявляти помилки й виконувати діагностику вебдодатку.

Зв’язок між сервером на Render та базою даних Aiven буде здійснюватись через захищене підключення за допомогою SSL-сертифікатів, що гарантує безпеку передачі даних. Облікові дані для підключення до бази будуть збережені в секретному сховищі Render (Environment Variables), звідки вони динамічно підставляються в процесі виконання серверного коду. Цей зв’язок буде реалізовано у вигляді стандартного MySQL-з’єднання в Node.js з використанням клієнта, який підтримує SSL-захист і підключення до віддалених баз. Такий

підхід дозволить відмовитись від локального або окремого сервера бази даних, зберігаючи гнучкість і контроль над архітектурою проєкту. Завдяки цим можливостям Render є оптимальним вибором для хостингу backend-частини невеликих і середніх вебпроєктів, зокрема у навчально орієнтованих розробках.

Під час проєктування архітектури вебзастосунку особливу увагу буде приділено справності усіх рівнів – від клієнтського інтерфейсу до серверної логіки та зберігання даних. Обраний стек технологій не лише відповідає сучасним вимогам до швидкодії, адаптивності та безпеки, а й органічно доповнює один одного. Завдяки використанню єдиної мови програмування для frontend і backend-частин зменшується складність самого проєкту та спрощується його підтримка у майбутньому. Це дозволить підтримувати єдиний підхід до обробки подій, дій, перевірки введених даних, формування відповідей API та взаємодії з базою даних.

У рамках реалізації вебзастосунку передбачено побудову гнучкої модульної структури, яка дозволить легко оновлювати або розширювати функціональність. Кожен логічний блок буде відповідати за окрему частину функціоналу – автентифікацію, роботу з навчальними матеріалами, тестування, формування сертифікатів тощо. Такий розподіл відповідальностей забезпечить незалежність, стабільність системи та дозволить уникати конфліктів при зміні окремих компонентів.

Обрана архітектура дозволить створити не просто робочий MVP-прототип, а повноцінну основу для подальшого удосконалення відповідно до нових функціональних потреб. Завдяки чіткому розмежуванню логіки між клієнтською та серверною частинами система залишатиметься зрозумілою й легко підтримуваною. Це дозволить оперативно вносити зміни, додавати нові можливості та адаптувати продукт до змін вимог без потреби в повній перебудові.

Висновки до розділу 2

У цьому розділі було здійснено загальний аналіз вимог до майбутнього вебзастосунку та обґрунтовано вибір інструментів, засобів і підходів до реалізації для вивчення англійської мови. Ретельно сформовані функціональні й нефункціональні вимоги стали основою для технічного проєктування системи, яке враховує потреби цільової аудиторії, сучасні стандарти у сфері UX/UI-дизайну, безпеки та масштабованості.

На основі аналізу предметної області та вивчення досвіду використання подібних платформ було сформовано чіткий перелік завдань, які має вирішувати вебдодаток. Обрана модифікована каскадна модель дозволила забезпечити гнучкість у плануванні етапів розробки та передбачити можливість повернення до попередніх стадій у разі зміни вимог. Визначення структури додатку на основі модульної архітектури забезпечує простоту масштабування та підтримки.

Загалом, результати цього розділу демонструють системний підхід до проєктування освітніх вебзастосунків, що базується на практичному досвіді, аналізі існуючих рішень, а також на сучасних практиках розробки. Проведена робота створює надійну основу для наступного етапу – реалізації вебдодатку відповідно до поставлених вимог.

РОЗДІЛ 3
РОЗРОБКА ВЕБДОДАТКУ ДЛЯ ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ

3.1 Практична реалізація об’єкта проєктування

Реалізація програмного продукту здійснювалася поступово, із чітким дотриманням вимог, сформованих у попередніх розділах. Основна увага приділялася не лише створенню функціоналу, але й структурованості, масштабованості, безпеці та адаптивності коду. Під час розробки було використано стек технологій, який дозволив забезпечити ефективну взаємодію між клієнтською, серверною та інформаційною частинами системи, реалізувати сучасний інтерфейс і надійне зберігання даних.

У процесі створення інтерфейсу вебдодатку було використано HTML5, як основу для структури сторінок, та SCSS – потужний CSS-препроцесор, що дозволив створити гнучкі, масштабовані стилі за допомогою вкладеності, змінних, міксинів і логічних операторів. Усі стилі структуровані за принципами BEM-методології, що підвищило читабельність і підтримуваність коду. SCSS-файли було організовано за секціями й імпортовано через головний style.scss.

Для оптимізації кінцевого файлу стилю вебдодатку було використано інструмент TailwindCSS, який є корисним CSS-фреймворком. Його головна ідея полягає в тому, щоб надати набір готових класів, що відображаються безпосередньо в HTML-розмітці і відповідають за стилізацію елементів. TailwindCSS не використовувався у даному проєкті як основний спосіб побудови інтерфейсу, оскільки структура стилів реалізована переважно за допомогою SCSS та методології BEM. Проте фреймворк виконав важливу роль у процесі фінальної оптимізації файлу стилів проєкту.

Було задіяно механізм purge (tree-shaking) [12], вбудований у TailwindCSS. Завдяки йому система автоматично аналізує HTML та JS-файли, виявляє, які класи дійсно використовуються в проєкті, і видаляє всі невикористані стилі з фінального CSS-файлу. У результаті цього вдалося значно зменшити обсяг

стилів, які передаються клієнту, що позитивно позначилося на швидкості рендеренгу і завантаження сторінок.

На фронтенді реалізовано такі основні сторінки: index.html, about.html, contact.html, course.html, learnProcess.html, ourMission.html. Кожна з яких має свою семантичну структуру, з чітким виділенням контентних блоків. Форми реалізовано з урахуванням правил валідації, placeholders та відповідними підказками для користувача.

Інтерактивна логіка, пов'язана з відкриттям і закриттям модальних вікон, обробкою форм, динамічною взаємодією з сервером, реалізована за допомогою JavaScript ES6+. Наприклад, у файлі modal.js (ліст. 3.1) описано логіку для відкриття модальних форм реєстрації й входу, обробку закриття за натисканням кнопки або поза межами вікна, блокування скролу під час відкриття.

Лістинг 3.1 – Демонстрація функціоналу роботи модального вікна

```
function openModal(modal) {
    if (!modal) return;
    modal.style.display = "flex";
    setTimeout(() => {
        modal.classList.add("modal--active");
        document.body.style.overflow = "hidden";
    }, 10);
}
function closeModal(modal) {
    if (!modal || !modal.classList.contains("modal--active")) return;
    modal.classList.add("modal--closing");
    setTimeout(() => {
        modal.classList.remove("modal--active", "modal--closing");
        modal.style.display = "none";
        document.body.style.overflow = "";
    }, 400);
}
```

кінець лістингу 3.1

Скрипт ContactForm.js (ліст. 3.2) відповідає за надсилання даних з форми зворотного зв'язку. Для обробки мережевих помилок застосовується конструкція try...catch, що дозволяє відображати користувачу повідомлення про збій в роботі у разі недоступності сервера або неправильного формату відповіді.

Лістинг 3.2 – Демонстрація роботи форми зворотнього зв’язку

```
        try {
            const response = await fetch("/send-message", {
                method: "POST",
                headers: {
                    "Content-Type": "application/json"
                },
                body: JSON.stringify(formData)
            });
            const result = await response.json();
            if (response.ok) {
                document.querySelector('input[placeholder="ПІБ"]').value
= "";

document.querySelector('input[placeholder="Email"]').value = "";
                document.querySelector('input[placeholder="Номер тел.
(Формат +380)"]').value = "";
                document.querySelector('input[placeholder="Тема"]').value
= "";
                document.querySelector('textarea[placeholder="Напишіть
повідомлення"]').value = "";
                sessionStorage.setItem("formSubmitted", "true");
                modal.style.display = "flex";
            } else {
                alert("Помилка при відправці: " + result.error);
            }
        } catch (error) {
            alert("Сталася помилка: " + error.message);
        }
    });
```

кінець лістингу 3.2

Для перевірки коректності API застосовувався Postman (рис. 3.1), який дозволив перевірити як правильність маршрутизації, так і поведінку сервера при помилкових запитах. Прикладом була перевірка з вже існуючим користувачем, яка повертає статус 200 та відповідне повідомлення у форматі JSON. У ході тестування були успішно перевірені як GET-, так і POST-запити до серверних маршрутів із різними комбінаціями параметрів. Це дозволило переконатися в стабільній роботі системи в умовах як очікуваних, так і неочікуваних нештатних сценаріїв.

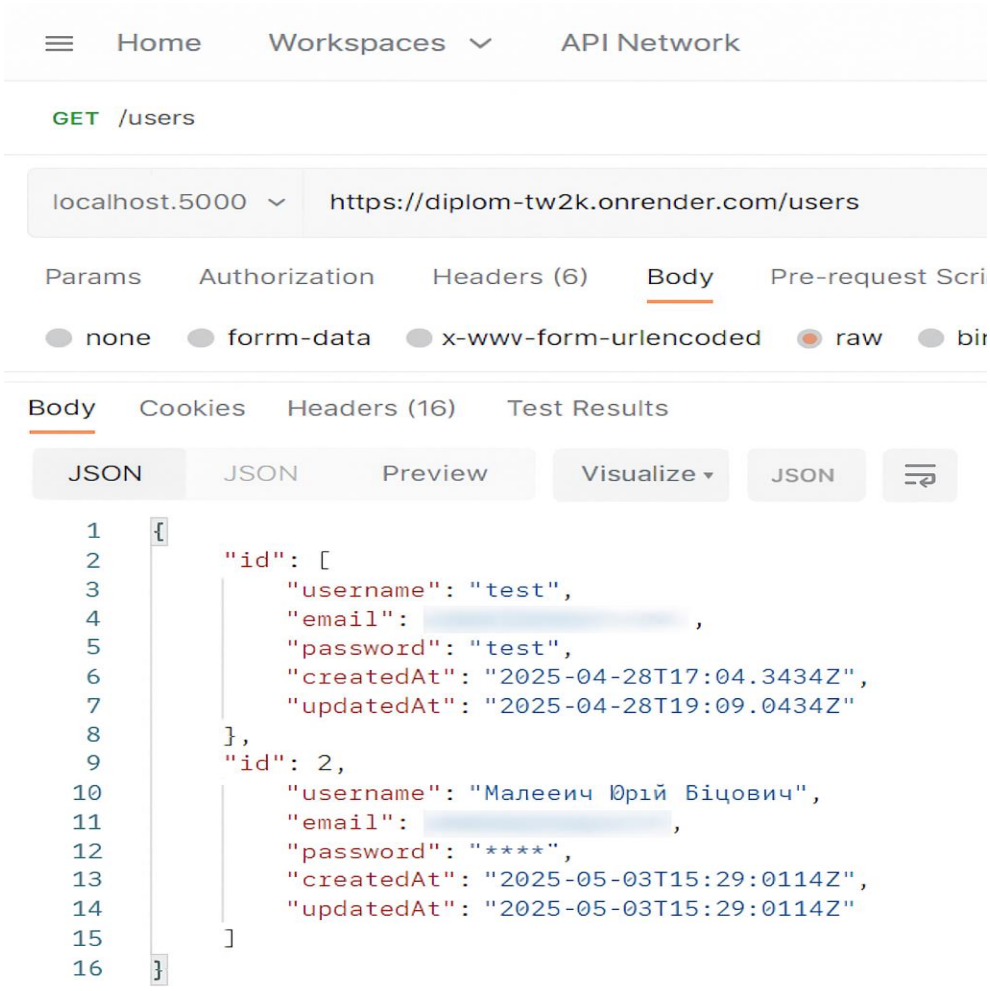

Рисунок 3.1 – Демонстрація справності роботи маршрутизації

Генерація сертифіката реалізована як окремий функціональний модуль, що активується після успішного проходження користувачем хоча б 1 блоку усіх етапів тестування. Сертифікат формується у форматі PDF (рис. 3.2) із зазначенням дати завершення навчання та підписом керівника проєкту. Всі текстові дані динамічно підставляються зі збережених результатів у базі даних MySQL. Генерація виконується безпосередньо на клієнтській стороні з використанням бібліотеки jsPDF, що дозволяє уникнути надмірного навантаження на сервер і забезпечити моментальне завантаження документа користувачем. Такий підхід спрощує процес перевірки результатів і створює додаткову мотивацію для завершення навчального курсу.

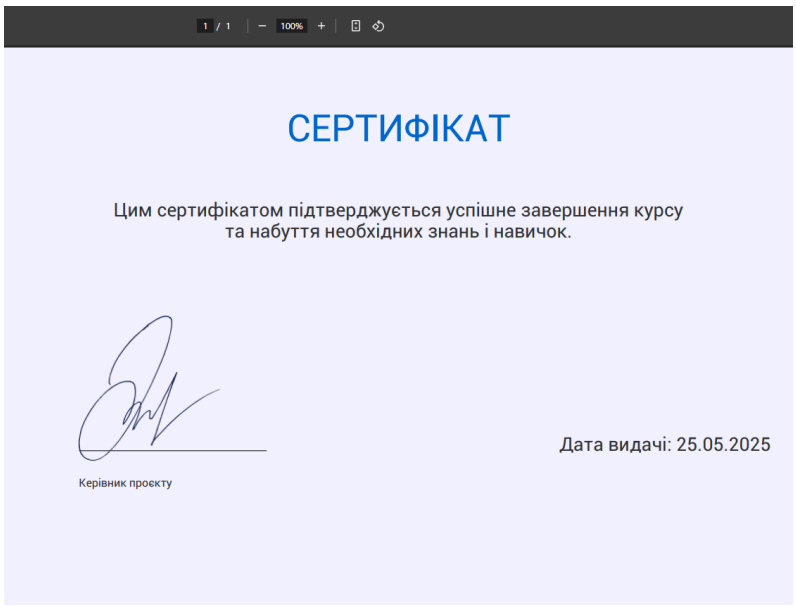

Рисунок 3.2 – Демонстрація отриманого сертифікату

Уся розробка велася у Visual Studio Code, із використанням розширень ESLint [13], Live Server [14], eCSStractor [15] (рис. 3.3). Ці інструменти забезпечили можливість перевірки в реальному часі, стабільність, читабельність та візуальну підтримку стилів у процесі написання.

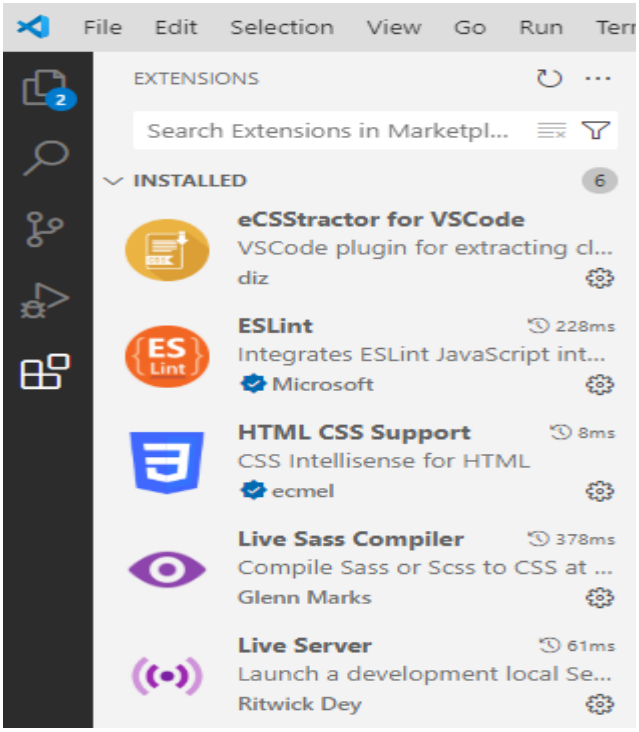

Рисунок 3.3 – Демонстрація встановлених розширень

У якості системи керування базами даних було обрано Aiven for MySQL – керовану хмарну технологію, яка забезпечує високий рівень надійності, масштабованості та безпеки зберігання даних. Підключення до бази здійснюється за допомогою ORM Sequelize (ліст. 3.3), що дозволяє працювати з таблицями на рівні об'єктів, що покращує підтримку і безпеку. Це дає змогу ефективно створювати, змінювати й зчитувати записи без написання сирих SQL-запитів.

Лістинг 3.3 – Підключення бази даних через ORM Sequelize

```
const { Sequelize, DataTypes } = require("sequelize");

const sequelize = new Sequelize(process.env.DATABASE_URL, {
    dialect: 'mysql',
    logging: false,
    dialectOptions: {
        ssl: {
            rejectUnauthorized: false
        }
    }
});
```

кінець лістингу 3.3

Саме таким чином, у межах даного етапу реалізації було створено частини вебдодатку. Усі ключові функції – інтерфейс сторінок, обробка форм, маршрутизація, робота з базою даних та оптимізація стилів – реалізовані з використанням сучасного і взаємодоповнюючого стеку технологій на основі аналізу предметної області.

Застосування SCSS забезпечило гнучке оформлення інтерфейсу, TailwindCSS – мінімізацію об’єму стилів, а React та Node.js – динамічну взаємодію між клієнтською і сервеною частинами. У результаті створено стабільний, адаптивний і масштабований додаток, готовий до подальшого тестування та розгортання.

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

Тестування та налагодження є завершальними, але надзвичайно важливими етапами життєвого циклу розробки програмного забезпечення. Саме вони дозволяють забезпечити відповідність створеної інформаційної системи до функціональних вимог, перевірити її стабільність, надійність, адаптивність й готовність до практичного використання кінцевими користувачами.

У межах реалізації цієї кваліфікаційної роботи цим етапам було приділено особливу увагу, оскільки вебзастосунок освітнього призначення, який має опрацьовувати особисті дані, забезпечувати логіку навчального процесу та демонструвати високу якість UX.

Тестування охоплювало клієнтську, серверну та базову частини вебзастосунку. Перевірка користувацького інтерфейсу проводилась вручну за допомогою браузерів Google Chrome і Mozilla Firefox із активним використанням інструментів розробника (DevTools). Це дозволило не лише аналізувати структуру сторінок, а й моделювати зміну розмірів для перевірки адаптивності відповідно до різних розмірів екрану, перевіряти швидкодію, якість верстки, зручність розміщення інтерактивних елементів.

Була протестована кожна функціональна частина вебдодатку: перевірено правильність відображення головної сторінки, курсу, сторінки «Про нас», форм зворотного зв’язку, модальних вікон. Було змодельовано ситуації з введенням коректних і некоректних даних, перевірено роботу форм, наявність підказок і повідомлень про помилки для користувачів. Особлива увага приділялась формі реєстрації, яка містить логіку перевірки email, пароля, уникнення дублювання даних.

Також було досліджено поведінку JavaScript-логіки в разі збоїв, разом з логікою блокування прокручування при відкритті модального вікна, автоматичного очищення форми після успішного відправлення даних, відображення повідомлень про помилку або підтвердження.

Серверна частина тестувалася із застосуванням утиліти Postman, яка дозволила створити запити до раніше створених маршрутів і перевірити їх відповідність очікуваному результату. Відправлялись GET і POST-запити до маршруту авторизації, реєстрації, надсилання форм. У всіх випадках перевірялась правильність обробки коректних і помилкових даних. Система давала відповідь відповідними статус-кодами HTTP, повертала повідомлення у форматі JSON з розгорнутими поясненнями. При цьому серверна логіка, реалізована через Node.js та Express, коректно обробляла несправності завдяки конструкціям try...catch, що забезпечило можливість швидко відловлювати помилки. Також тестувалась логіка взаємодії з базою даних: створення нових користувачів, обробка дублювань значень, перевірка унікальності, збереження сесій.

Особливу увагу приділено взаємодії з хмарною базою даних Aiven for MySQL, підключення до якої здійснюється через ORM Sequelize. Це включає в себе перевірку унікальності записів, коректність типів даних, відповідність обов’язкових полів та контроль транзакцій. Такий підхід дозволив досягти стабільної роботи backend-частини, знизити ризик втрати даних, а також підвищити загальну стійкість додатку до непередбачуваних ситуацій, пов’язаних із підключенням або обробкою запитів. У процесі тестування перевірялось не лише збереження нових записів, але й реакція системи на некоректні запити, неправильну структуру об'єктів, відсутність підключення до бази. Усі помилки було виявлено і усунено в процесі налагодження.

За допомогою DevTools і вкладки Network в браузері вивчався можливий трафік, час відповіді, структура JSON-відповідей, а також поведінка сторінки у випадку довгого очікування або втрати підключення від користувача.

Оцінка адаптивності інтерфейсу проводилась шляхом емуляції поведінки додатку на різних пристроях: мобільних телефонах, планшетах, ноутбуках, декстопах (рис. 3.4). Було перевірено збереження пропорцій, розташування блоків, читабельність тексту та доступність елементів керування при зміні ширини вікна.

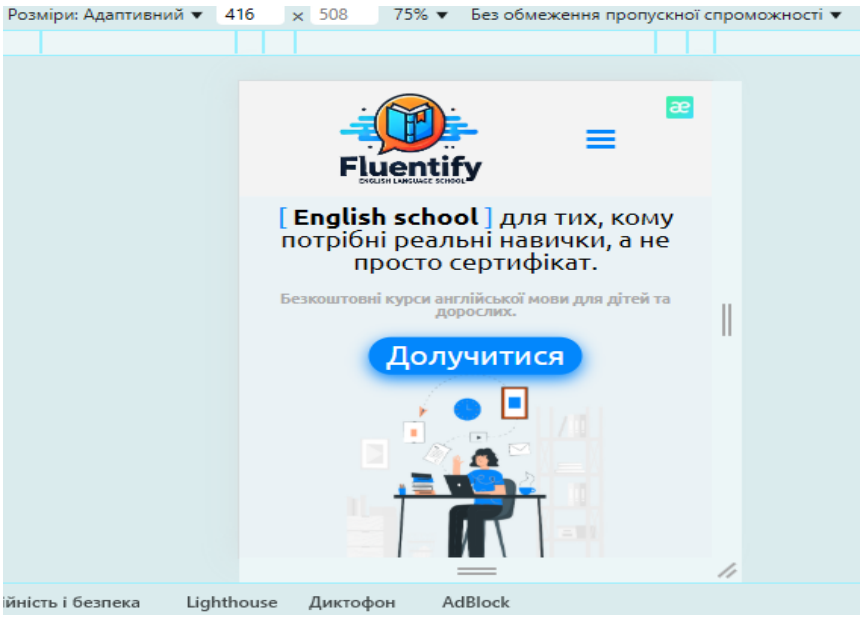

Рисунок 3.4 – Демонстрація адаптивного варіанту сайту

Візуальна структура залишалась чіткою, не спотворювалась, усі елементи були адаптовані до змін розміру вікна за допомогою медіа запитів CSS. Перевірка продуктивності показала ефективність використання TailwindCSS, зокрема його механізму purge, завдяки чому розмір фінального CSS-файлу був суттєво зменшений, що сприяло швидкому завантаженню сторінок навіть при повільному інтернет-з'єднанні (рис. 3.5). Цей підхід дозволив забезпечити стабільну взаємодію між клієнтською та серверною частинами додатку на кожному етапі розробки.

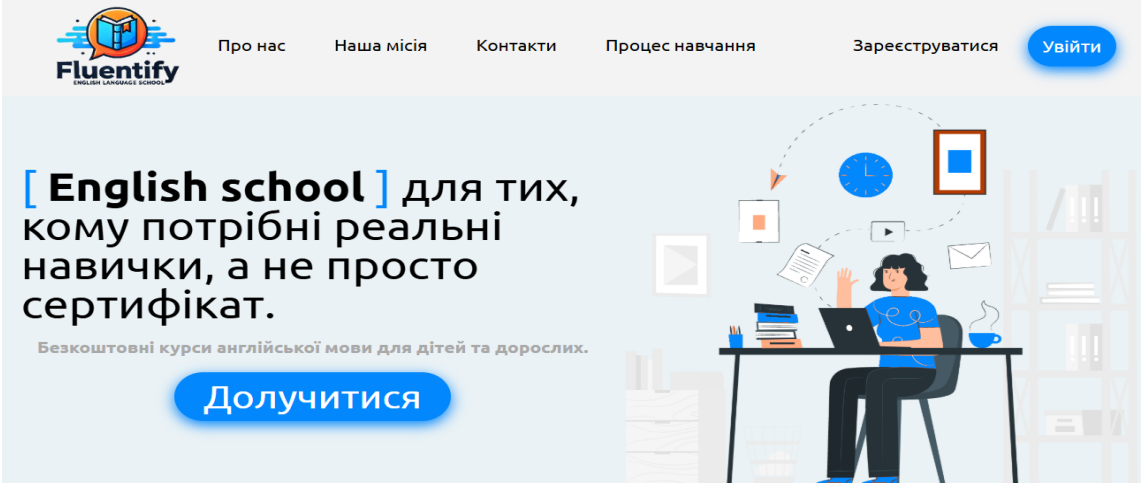

Рисунок 3.5 – Зображення головної сторінки

Було проведено ручне тестування логіки входу в систему, збереження сесій, відображення стану авторизації у клієнтській частині. Система відпрацьовувала сценарії введення неправильного пароля, неіснуючого логіна, повторної реєстрації з тією ж адресою – у кожному випадку повертались відповідні помилки (рис. 3.6) з поясненням, що гарантує прозору взаємодію з користувачем.

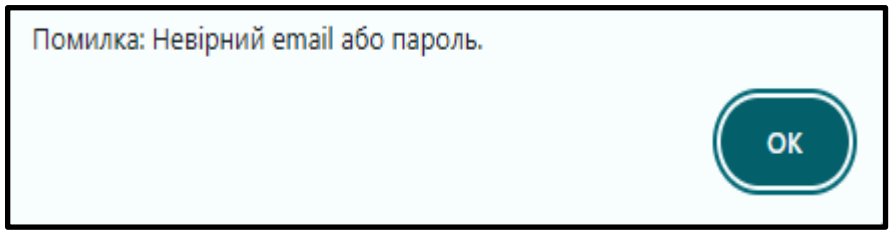

Рисунок 3.6 – Помилка авторизації користувача

Налагодження здійснювалося поетапно. Під час реалізації логіки активно використовувались засоби логування (console.log()), дебагер у середовищі розробки Visual Studio Code, а також можливості логування на хостингу Render (рис. 3.7).

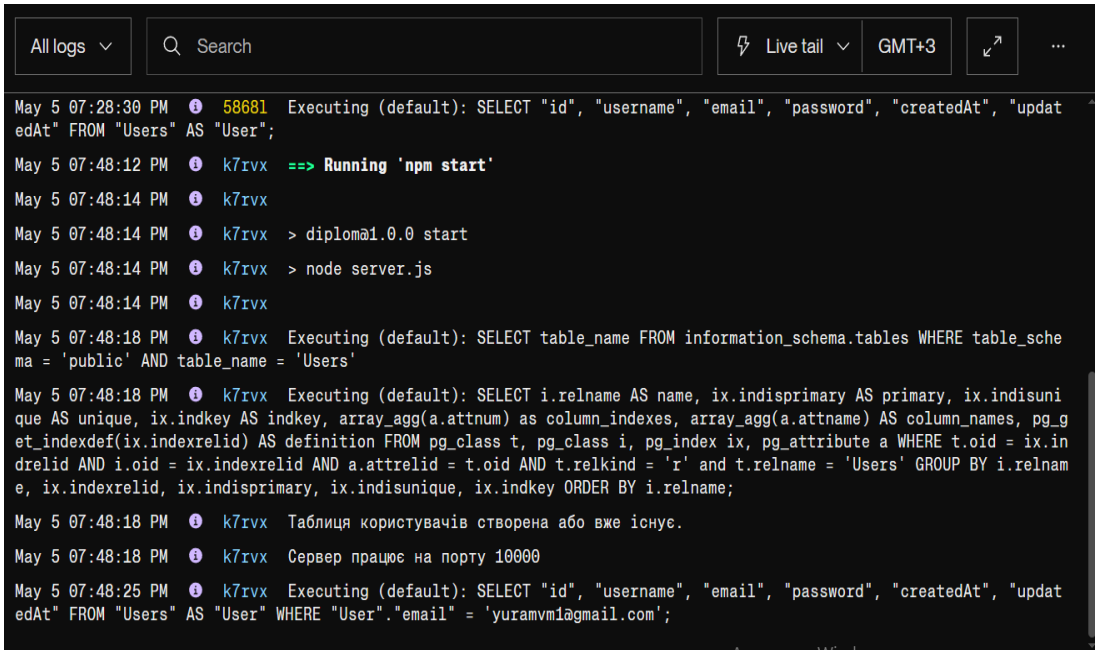

Рисунок 3.7 – Логування Render у реальному часі

Це дозволило швидко виявляти помилки у серверній частині, реагувати на невірні запити та аналізувати поведінку системи в різних ситуаціях у реальному часі. Окремо перевірялась безпека передачі даних, коректність зберігання змінних середовища, а також ізоляція облікових даних через механізми середовища виконання коду. Усі конфіденційні параметри зберігалися у .env-файлі й не потрапляли до публічного репозиторію з огляду на безпеку.

За результатами тестування було підтверджено, що система є повністю функціональною, стійкою до типових помилок, здатною до масштабування та демонструє стабільну роботу в умовах реального використання. Виявлені недоліки були усунені, а розроблене рішення пройшло перевірку відповідності всім технічним, функціональним і нефункціональним критеріям, визначеним у завданні до кваліфікаційної роботи. Вебдодаток готовий до подальшого розгортання та використання у навчальному середовищі.

3.3 Розробка інсталяційного пакета

Інсталяційний пакет є важливою частиною розробки програмного забезпечення, оскільки саме він забезпечує можливість розгортання, налаштування та запуску вебдодатку в середовищі. У контексті даного проєкту для вивчення англійської мови – інсталяційний пакет охоплює весь набір файлів, конфігурацій та інструкцій, необхідних для коректного запуску клієнтської та серверної частин, а також підключення до бази даних. Усі елементи інсталяції структуровано так, щоб забезпечити розгортання системи як у хмарному середовищі, так і локально.

Умовно інсталяційний процес можна розділити на три частини: підготовка клієнтської частини, налаштування серверної логіки та організація підключення до бази даних. Клієнтська частина проєкту, розроблена з використанням React, має власну структуру – папки, конфігураційні файли, а також інструкції для складання білду на хостингу. Після компіляції фронтенд створює оптимізований

набір HTML, CSS і JS-файлів у директорії, який згодом може бути або передано на хостинг, або з’єднано з Express-сервером.

Серверна частина побудована на основі Node.js із використанням Express.js, що дозволяє створити API для обробки клієнтських запитів, взаємодії з базою даних та логіки авторизації користувача. Основні конфігураційні файли включають основний вхідний скрипт, роути, .env для налаштування змінних середовища (порт, облікові дані БД), та package.json з усіма залежностями. Для інсталяції серверної частини достатньо виконати команду npm install у відповідному каталозі, після чого система буде готова до запуску.

На хостингу доступна інструкція для розгортання, у якій поетапно описано: налаштування середовища Node.js, встановлення залежностей, запуск фронтенд частини в режимі розробника, складання повного білду, запуск серверної частини, приклади змінних середовища, а також можливі помилки та способи їх усунення.

Окремо реалізовано налаштування для автоматичного розгортання в хмарному середовищі Render. У репозиторії проєкту зберігається інформація про структуру сторінок, вказується вхідний файл серверу, версія Node.js, а також скрипт для білду фронтенду, що автоматично запускається перед публікацією на хостинг. Завдяки підключенню репозиторію до GitHub та налаштуванню гілки для деплою, кожен новий коміт автоматично ініціює збірку проєкту та оновлення на сервері. Це дозволяє значно спростити обслуговування додатку та усунути людський фактор при оновленні версії продукту.

Підключення до бази даних реалізується через окремий сервіс Aiven. Для цього використовується стандартний MySQL-клієнт для Node.js, а також .env-файл із відповідними ключами: DB_HOST, DB_USER, DB_PASSWORD, DB_NAME. Ці параметри зчитуються при запуску сервера та використовуються для встановлення з’єднання з клієнтом. Також додаються SSL-параметри або окремі конфігураційні шляхи для сертифікатів. Завдяки такому підходу база може бути легко змінена або перенесена, без зміни коду.

Для локального розгортання передбачена спрощена схема запуску. Встановлюється Node.js, після чого можна починати роботу. За замовчуванням інтерфейс доступний для локального відображення. Це дозволяє розгортати систему не лише на хостингу, але й для локального демонстрування або тестування на інших пристроях у локальній мережі.

У результаті сформовано повноцінний інсталяційний пакет, що охоплює як середовище розробки, так і хмарну інфраструктуру для запуску. Такий підхід дозволяє не лише швидко публікувати нові версії, а й забезпечити впровадження вебдодатку у реальні умови з мінімальними затратами часу та ресурсів.

3.4 Захист інформаційно-комп’ютерної системи

Забезпечення інформаційної безпеки є ключовим фактором при розробці сучасних вебдодатків, особливо у сфері, де відбувається обробка персональних даних користувачів. Навіть якщо в рамках кваліфікаційної роботи функціонал є демонстраційним, уся архітектура проєкту була побудована з урахуванням актуальних вимог до безпеки вебзастосунків. Основна увага приділялася захисту передачі даних, структурі серверної логіки і надійності підключення до бази даних.

Одним із найкращих рішень було застосування захищеного протоколу передачі даних при підключенні до бази даних Aiven for MySQL. Це підключення реалізовано з використанням SSL-сертифікатів, що забезпечує шифрування усіх запитів, а також захищає від перехоплення даних під час їхньої передачі між клієнтом і сервером. Облікові дані для доступу до бази не зберігаються у відкритому вигляді в коді проєкту, а виносяться до сховища середовища виконання Render (Environment Variables) (рис. 3.8), звідки вони динамічно підставляються в момент запуску. Такий підхід дозволяє запобігти витоку критичних ключів навіть у випадку втрати доступу до коду чи неправильного деплою.

Рисунок 3.8 – Надійне збереження облікових даних для доступу до БД

Особливу увагу приділено захисту від SQL-ін’єкцій. Оскільки вся взаємодія з базою реалізована через ORM Sequelize, система автоматично використовує підготовлені запити (prepared statements), які надійно екранують усі введені дані. Це значно знижує ризик виконання довільного SQL-коду та забезпечує безпеку при зчитуванні, оновленні чи записі інформації до бази даних.

На рівні клієнтської частини було реалізовано базові заходи захисту від міжсайтового скриптового впровадження коду (XSS). Зокрема, при виведенні даних у DOM не використовуються потенційно небезпечні методи, як innerHTML. Усі введені дані піддаються фільтрації, що дає змогу запобігти виконанню сторонніх скриптів, які могли б бути впроваджені через поля введення користувача.

З метою підвищення надійності система була розгорнута у хмарному середовищі, що підтримує резервне копіювання. Такий підхід забезпечує збереження даних навіть у випадку збою або втрати доступу до сервера. Потрібно зазначити, що інсталяція даного вебдодатку відрізняється від традиційної для десктопних систем: вона не потребує запуску окремого інсталятора, а здійснюється через клонування репозиторію з GitHub, встановлення залежностей за допомогою стандартної команди npm install та запуск сервера клієнта.

Структура вебдодатку є чітко організованою та логічно розділеною на окремі модулі. Клієнтська частина містить HTML-шаблони, стилі, написані з використанням SaSS, та JavaScript-скрипти, що забезпечують взаємодію з користувачем, обробку форм та інші елементи інтерфейсу. Серверна частина реалізована на основі Node.js з фреймворком Express і включає маршрутизацію запитів, логіку обробки даних, зв’язок із базою MySQL (через ORM Sequelize), а також сервіси надсилання повідомлень. Це дозволяє гарантувати безпеку при розгортанні та експлуатації системи.

3.5 Аналіз гнучкості та масштабованості системи

У сучасному програмному забезпеченні гнучкість і масштабованість системи виступають критично важливими характеристиками, що забезпечують її стійкість до змін, здатність до розвитку та ефективне функціонування в умовах зростання навантажень на клієнт. Розроблений у цій кваліфікаційній роботі вебдодаток має низку архітектурних і технологічних рішень, які дають змогу розглядати його як рішення, придатне для подальшого масштабування без потреби принципових змін внутрішньої структури чи загальної логіки функціонування модулів.

Одним із ключових факторів гнучкості є модульність архітектури. Кожен функціональний блок – реєстрація, авторизація, навчальні модулі, тестування, формування сертифіката, тощо – реалізовано окремо (рис. 3.9), що забезпечує ізоляцію змін та зручність у майбутній підтримці. За необхідності можна змінити або доповнити окремий модуль, не порушуючи роботу всієї системи. Це особливо важливо у випадку розширення функціоналу або адаптації системи під нові можливі потреби користувачів. У випадку розробки – це полегшує навігацію в файлах системи. При необхідності можливо легко віднайти потрібний файл, оскільки кожен окремий модуль підписано.

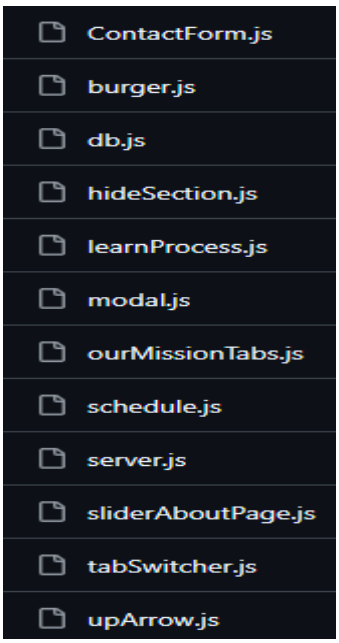

Рисунок 3.9 – Зображення функціональних блоків

Система має чітко окреслену файлову структуру (рис. 3.10), у якій компоненти логічно розділені за призначенням: сторінки, стилі, маршрути, логіка обробки запитів, підключення до бази даних. Це спрощує навігацію по проєкту, пришвидшує пошук необхідного функціоналу та забезпечує легкість масштабування.

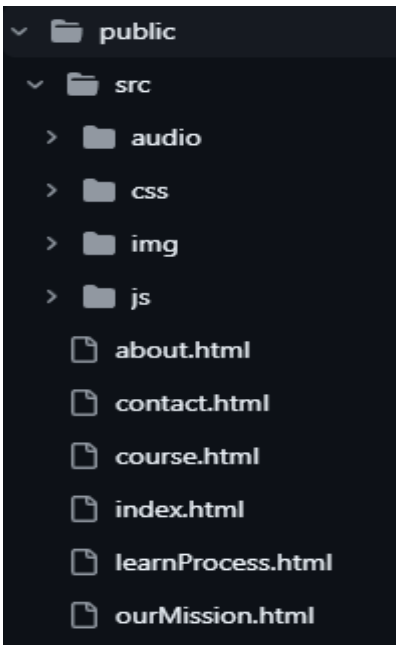

Рисунок 3.10 – Файлова структура проєкту

З точки зору масштабованості, вебдодаток підтримує можливість обробки зростаючої кількості користувачів і навчального контенту. Завдяки використанню реляційної бази даних (MySQL), структура таблиць може бути доповнена новими атрибутами або зв’язками без потреби у кардинальному рефакторингу коду. Крім того, у випадку можливого переходу на інші хмарні сервіси, система не потребує глибокої переробки коду, оскільки взаємодія з БД відбувається через ORM Sequelize, що забезпечує гнучкість у зміні типу СУБД.

Архітектура побудована з урахуванням майбутньої інтеграції з іншими сервісами. Наприклад, система може бути доповнена новими API для зовнішніх платформ, підтримкою додаткової авторизації, або розширена за рахунок додавання окремих сервісів, модулів.

Окремо варто підкреслити, що система має потенціал для локалізації. Хоча наразі реалізовано лише україномовний інтерфейс, структура дозволяє легко додати підтримку інших мов через механізм шаблонізації, плагінів та текстових словників. Це важливо при масштабуванні на нові регіони або при інтеграції у платформи, орієнтовані на багатомовну аудиторію.

Важливим аспектом масштабованості є й те, як система реагує на зростання навантаження. У межах кваліфікаційної роботи передбачались базові передумови для цього. Наприклад, можливість масштабування серверної частини, використання кешування, застосування CDN для статичних ресурсів – усе це може бути реалізовано у наступних ітераціях розвитку продукту без потреби в його перебудові з нуля.

Загалом, архітектурні рішення, прийняті під час реалізації даного вебзастосунку, демонструють здатність системи адаптуватися до нових можливих функціональних вимог, змін навантаження та потреб цільової аудиторії. Це забезпечує її довгострокову життєздатність та придатність до використання у реальних умовах, включно з подальшим масштабуванням у рамках освітніх платформ, інтеграцією з зовнішніми системами або запуском у різних регіонах з відповідною локалізацією. Такий рівень гнучкості є важливим

фактором якісної інженерної реалізації та здатності системи до еволюції відповідно до змін ринку та технологічного середовища.

Висновки до розділу 3

У третьому розділі було реалізовано повноцінний вебдодаток для вивчення англійської мови, що поєднує клієнтську, серверну та інформаційну складові. При розробці інтерфейсу використано HTML5, JavaScript, SCSS із методологією BEM, а також TailwindCSS для оптимізації стилів.

У процесі тестування була перевірена працездатність усіх функціональних компонентів: обробка форм, маршрутизація запитів, авторизація, валідація форм, робота з базою даних та обробка помилок. Клієнтський інтерфейс успішно пройшов перевірку адаптивності, швидкодії та зручності використання на різних типах пристроїв навіть з слабким інтернет з’єднанням. Тестування охоплювало і стабільність взаємодії з сервером, правильність відповіді API, роботу у випадках відсутності з’єднання чи введення некоректних даних.

Значну увагу приділено питанням безпеки. Система захищена від міжсайтових скриптових атак і SQL-ін’єкцій, а передача даних здійснюється через захищене SSL-з’єднання. Облікові дані зберігаються у захищеному середовищі середовища виконання, що унеможливлює їх випадкове потрапляння до публічного репозиторію. Проєкт розгорнуто на хмарній платформі Render з автоматизованим деплоєм, що забезпечує зручність в оновленні та масштабуванні.

Можна стверджувати, що реалізований вебдодаток відповідає всім поставленим вимогам щодо функціональності, надійності, адаптивності та безпеки. Обрана архітектура виявилася достатньо гнучкою для подальшого розширення, інтеграції з іншими сервісами, додавання нових функцій чи мовних локалізацій. Це дозволяє вважати розроблену систему придатною до практичного застосування в освітньому середовищі.

ВИСНОВКИ

У процесі реалізації проєкту було здійснено аналіз сучасних рішень у сфері онлайн-навчання, зокрема вебплатформ для вивчення англійської мови.

Було проаналізовано функціональні можливості таких сервісів, як Duolingo, Babbel, BBC Learning English та інших, виявлено їхні обмеження – відсутність персоналізації, закритість архітектури, неповна локалізація тощо. Це дозволило чітко сформулювати напрямки вдосконалення та визначити функціональну модель власного продукту.

На основі виявлених недоліків та цілей розробки були сформульовані вимоги до майбутньої системи. Основними з них стали: наявність україномовного інтерфейсу, підтримка тем із граматики, лексики, тестування, фіксація результатів користувача, генерація сертифіката, а також адаптивна верстка для різних пристроїв. Визначення вимог стало основою для побудови подальшої архітектури та вибору технологій.

Згідно з поставленими вимогами, було обґрунтовано вибір технологічного стеку: React – для побудови динамічного інтерфейсу, SCSS і TailwindCSS – для ефективного стилізування, Node.js та Express – для обробки запитів на сервері, MySQL – як СУБД для зберігання даних користувачів. Обрані інструменти забезпечують відповідність вимогам продуктивності, безпеки, розширюваності та стабільності системи.

Архітектуру системи спроєктовано за трирівневою моделлю: клієнтська частина, сервер та база даних. Усі рівні взаємодіють через API, що дозволяє ізолювати логіку обробки даних, інтерфейс та зберігання. Така структура забезпечує зрозумілу взаємодію між компонентами й полегшує майбутнє масштабування та супровід.

У межах розробки було реалізовано основні функціональні модулі: особистий кабінет, сторінки навчання, тестування, збереження результатів і генерація PDF-сертифіката. Всі модулі організовано відповідно до

компонентного підходу, що дозволяє легко змінювати, доповнювати або повторно використовувати частини інтерфейсу та логіки.

Особливу увагу приділено системі авторизації. Було реалізовано механізм реєстрації, входу, перевірки пароля, обробки дублюючих акаунтів, збереження сесій, а також контролю доступу до захищених маршрутів. Валідація даних та обробка помилок забезпечують стабільну й передбачувану взаємодію з користувачем.

Після реалізації основних модулів було проведено ручне тестування, а також перевірка API через Postman. Перевірено маршрути, сценарії некоректних запитів, помилкової авторизації, повторної реєстрації. Крім того, протестовано адаптивність інтерфейсу, обробку даних, оновлення інтерфейсу після входу та виходу користувача.

На завершальному етапі систему було розгорнуто на платформі Render. Налаштовано зв’язок із віддаленою базою даних Aiven for MySQL, збереження змінних середовища, запуск бекенду з автодеплоєм із GitHub. Завдяки цьому вебдодаток працює стабільно в хмарному середовищі та доступний для використання будь-де через браузер.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Quick Start – React. URL: https://react.dev/learn (date of access: 27.02.2025).

2. Sass: Documentation. Sass: Syntactically Awesome Style Sheets. URL: https://sass-lang.com/documentation/ (date of access: 02.03.2025).

3. Documentation – Tailwind CSS. Tailwind CSS – Rapidly build modern websites without ever leaving your HTML. URL: https://v2.tailwindcss.com/docs (date of access: 03.03.2025).

4. Index. Node.js v24.1.0 Documentation. Node.js – Run JavaScript Everywhere. URL: https://nodejs.org/docs/latest/api/ (date of access: 06.03.2025).

5. Express routing. Express – Node.js web application framework. URL: https://expressjs.com/en/guide/routing.html (date of access: 07.03.2025).

6. MySQL. MySQL Documentation. URL: https://dev.mysql.com/doc/ (date of access: 09.03.2025).

7. Aiven for MySQL®. Aiven docs. Aiven – Your AI-ready Open Source Data Platform. URL: https://aiven.io/docs/products/mysql (date of access: 11.03.2025).

8. Microsoft. Documentation for Visual Studio Code. Visual Studio Code – Code Editing. Redefined. URL: https://code.visualstudio.com/docs (date of access: 13.03.2025).

9. Git – Documentation. Git. URL: https://git-scm.com/doc (date of access: 13.03.2025).

10. Postman Docs. URL: https://learning.postman.com/docs/introduction/overview/ (date of access: 15.03.2025).

11. Docs + Quickstarts. Render. Cloud Application Platform. Render. URL: https://render.com/docs (date of access: 26.03.2025).

12. Tree shaking and code removal. Expo Documentation. URL: https://docs.expo.dev/guides/tree-shaking/ (date of access: 11.03.2025).

13. Documentation – ESLint – Pluggable JavaScript Linter. Find and fix problems in your JavaScript code – ESLint – Pluggable JavaScript Linter. URL: https://eslint.org/docs/latest/ (date of access: 13.03.2025).

14. Live Server – Visual Studio Marketplace. Extensions for Visual Studio family of products. Visual Studio Marketplace. URL: https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer (date of access: 13.03.2025).

15. GitHub – hudochenkov/ecsstractor. Text plugin for extracting class names from HTML and generate CSS stylesheet for following work. GitHub. URL: https://github.com/hudochenkov/ecsstractor (date of access: 13.03.2025).

ДОДАТКИ

Додаток А - Апробація

Міністерство освіти і науки України
Волинська обласна рада
Луцька міська рада
Луцький національний технічний університет
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
Національний університет «Львівська політехніка»
Військовий інститут телекомунікацій та інформатизації імені Героїв Крут (м. Київ)
Тернопільський національний педагогічний університет імені В. Гнатюка
Рівненський державний гуманітарний університет
Навчально-науковий інститут «Українська інженерно-педагогічна академія» Харківського національного університету ім. В.Н. Каразіна
Люблінська політехніка (Польща)
Фрайберзька гірнича академія (Німеччина)
Гронінгенський університет (Нідерланди)
Кавказький університет (Грузія)
Політехнічний університет Браганси (Португалія)

ITOHB

ТЕЗИ ДОПОВІДЕЙ
X Міжнародної науково-практичної конференції з проблем вищої освіти і науки «Інформаційні технології в освіті, науці і виробництві (ІТОНВ-2025)

23-24 травня 2025 р.

Луцьк 2025

УДК 004.4

УДОСКОНАЛЕННЯ ПРОЦЕСУ ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ ЗА ДОПОМОГОЮ ВЕБЗАСТОСУНКУ З БАЗОЮ ДАНИХ ТА ОСОБИСТИМ КАБІНЕТОМ КОРИСТУВАЧА

Ліщина Наталія Миколаївна
Луцький національний технічний університет, доцент кафедри інженерії програмного забезпечення, n.lishchyna@lutsk-ntu.com.ua
Малевич Юрій Валерійович
Луцький національний технічний університет, студент групи ІПЗ-41, yuramvm1@gmail.com

IMPROVING THE PROCESS OF LEARNING ENGLISH THROUGH A WEB-BASED APPLICATION WITH A DATABASE AND A USER ACCOUNT

Nataliia Lishchyna
Lutsk National Technical University, Associate Professor at the Department of Software Engineering, n.lishchyna@lutsk-ntu.com.ua
Yurii Malevych
Lutsk National Technical University, Student of the Group IPZ-41, yuramvm1@gmail.com

The paper discusses the expediency of introducing a web-based application aimed at optimizing the process of learning English. The system integrates a structured educational program with a personal user account and database, enabling learners to systematically progress through vocabulary, grammar, speaking and listening exercises, and track their results in real time.

Keywords: *education, English, web application, database, user account, testing, vocabulary, grammar.*

Активний розвиток цифрових технологій створює передумови для переосмислення підходів до організації освітнього процесу [1]. Зокрема, вивчення англійської мови, яке традиційно ґрунтується на статичних підручниках і фрагментарних онлайн-ресурсах, дедалі більше потребує інтегрованих цифрових рішень. У такому контексті доцільним є впровадження вебзастосунку, який поєднує навчальні матеріали, систему тестування, базу даних користувачів і функціонал особистого кабінету.

Доцільність розробки такого застосунку обумовлюється потребою в персоналізації навчального процесу. Класичні форми навчання не враховують індивідуальний темп засвоєння матеріалу, рівень знань чи пріоритетні напрями розвитку навичок. Завдяки особистому кабінету користувач отримує змогу відстежувати власні результати, повертатися до складних тем, отримувати автоматичні рекомендації щодо повторення та бачити динаміку прогресу. Це значною мірою підвищує мотивацію та ефективність навчання, а також сприяє формуванню довготривалого інтересу до освітнього процесу [1].

Централізоване збереження даних у базі дозволяє забезпечити безперервність та гнучкість освітнього процесу. Результати тестів, оцінки, виконані завдання та інші активності не зникають з часом і доступні з будь-якого пристрою. Це дає можливість не лише аналізувати індивідуальні досягнення, а й адаптувати вміст під потреби конкретного користувача або групи. Крім того, можливість збереження даних в історії дозволяє викладачам та адміністраторам системи оперативно відслідковувати динаміку успішності та приймати обґрунтовані педагогічні рішення [2].

Суттєвим фактором доцільності впровадження вебзастосунку є його здатність покращувати дисципліну та регулярність навчання. Завдяки автоматичним нагадуванням, щоденним завданням і системі досягнень, користувач отримує не лише доступ до матеріалу, а й структурований шлях його засвоєння. Це дозволяє уникати ситуацій, коли учень відкладає вивчення на невизначений термін. Введення таких інструментів, як система балів, прогрес-бари та візуальні цілі, сприяє формуванню внутрішньої мотивації та самоконтролю, що особливо актуально для учнів старших класів, студентів і дорослих користувачів.

Ще одним переконливим аргументом на користь створення подібного вебзастосунку є можливість глибокої аналітики навчального процесу. Система може збирати та обробляти дані про те, які завдання викликають найбільше труднощів, які теми найчастіше пропускаються, скільки часу витрачається на певні

блоки тощо. Це відкриває нові перспективи для побудови адаптивного навчання – такого, яке в реальному часі змінюється відповідно до потреб і поведінки користувача. У довгостроковій перспективі ці дані також можуть використовуватись для вдосконалення освітньої програми, прийняття рішень про оновлення матеріалів або створення нових курсів.

З методичної точки зору, система також дає змогу забезпечити структурованість і контрольованість навчального контенту. Розділи на зразок «Vocabulary», «Grammar», «Listening», «Speaking», доповнені тестами, створюють логічну послідовність засвоєння матеріалу. Водночас, аналітична система дозволяє виявляти прогалини у знаннях та коригувати навчальний маршрут без втручання викладача, що відкриває можливості для самостійного та гнучкого навчання.

Важливо також підкреслити масштабованість та гнучкість рішення. Такий застосунок може бути використаний не лише індивідуальними користувачами, а й навчальними закладами чи мовними школами. У перспективі можливе розширення функціоналу за рахунок інтеграції голосового розпізнавання, гейміфікації навчального процесу, створення рейтингів і досягнень, а також впровадження мультимедійних модулів із відео та аудіо для глибшого занурення у мовне середовище.

Таким чином, створення вебзастосунку для вивчення є доцільним, актуальним і стратегічно виправданим кроком, що відповідає сучасним освітнім тенденціям. Такий підхід забезпечує індивідуалізацію навчання, підвищення мотивації, аналітичну підтримку, технічну стабільність і потенціал до подальшого розвитку.

Список використаних джерел

1. Coursera Blog | Online learning news for learners, educators, & employers. *Coursera Blog*. URL: https://blog.coursera.org/ (date of access: 30.04.2025).
2. MDN Web Docs. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/ (date of access: 01.05.2025).