Міністерство освіти і науки України

Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему: «Проєктування та розробка сайту школи іноземних мов»

Виконала: студентка групи ІПЗ21-1
Спеціальність: 121 Інженерія програмного забезпечення

Дмитрієва А. О.
(прізвище та ініціали)

Керівник: к. фіз.-мат. н, доц. Лебідь О. Ю.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент: ДМСУ, Спеціалізоване управління розробки та супроводження програмного забезпечення, Департамент з питань цифрового розвитку, цифрових трансформацій та цифровізації
(місце роботи)
головний державний інспектор відділу розробки програмного забезпечення
(посада)
Бахтін О. В.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2025

АНОТАЦІЯ

Дмитрієва А. О. Проєктування та розробка сайту школи іноземних мов.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 121 «Інженерія програмного забезпечення». – Університет митної справи та фінансів, Дніпро, 2025.

Дана кваліфікаційна робота присвячена проєктуванню та розробці веб-застосунка для онлайн-школи іноземних мов. У роботі розглянуто сучасні підходи до створення освітніх вебплатформ, проаналізовано функціональні та нефункціональні вимоги до системи, а також обґрунтовано вибір технологічного стеку, що включає бібліотеку React для розробки клієнтської частини, та платформу Firebase (Firestore, Authentication, Hosting) для реалізації серверної логіки, бази даних та автентифікації. Проведено детальне проєктування архітектури системи та моделі даних, розроблено користувацький інтерфейс з акцентом на адаптивність та зручність використання.

Результатом роботи є функціональний, зручний та ефективний вебсайт онлайн школи вивчення іноземних мов, який відповідає сучасним вимогам користувачів та забезпечує потреби школи. Отримані результати мають практичне значення, оскільки розроблений вебзастосунок є готовим до впровадження рішенням, що забезпечує сучасний та ефективний інструмент для організації дистанційного навчання іноземним мовам.

Ключові слова: вебзастосунок, вебсайт, онлайн-школа, React, Firebase, Firebase Authentification, LiqPay, адаптивний дизайн.

Публікації:

1. Дмитрієва А. О., Костенко В. В. Using the library WinUI 3.0 to implement a unified interface based on the .NET 6.0 platform // Інноваційні технології, моделі управління кібербезпекою : матеріали IV Міжнар. наук. конф. (м. Дніпро, 15–17 квіт. 2024 р.). Дніпро : Університет митної справи та фінансів, 2024.

ABSTRACT

Dmytriieva A. O. Design and Development of a Website for a Foreign Language School.

Qualification thesis for obtaining a bachelor's degree in specialty 121 "Software Engineering". – University of Customs and Finance, Dnipro, 2025.

This qualification work is dedicated to the design and development of a web application for an online foreign language school. The paper explores modern approaches to building educational web platforms, analyzes the functional and non-functional requirements of the system, and justifies the choice of the technology stack, which includes the React library for frontend development and the Firebase platform (Firestore, Authentication, Hosting) for implementing server-side logic, database management, and authentication.

The system architecture and data model were carefully designed, and a user interface was developed with a focus on responsiveness and ease of use.

The result of this work is a functional, user-friendly, and efficient website for a foreign language online school, which meets modern user expectations and addresses the needs of the school. The outcomes of this project have practical significance, as the developed web application is ready for deployment and provides a modern and effective tool for organizing remote foreign language education.

Keywords: web application, website, online school, React, Firebase, Firebase Authentication, LiqPay, responsive design.

Publications:

1. Dmytriieva, A. O., & Kostenko, V. V. (2024). Using the library WinUI 3.0 to implement a unified interface based on the .NET 6.0 platform. Innovative technologies, models of cybersecurity management: Proceedings of the 4th International Scientific Conference (2024. 15.04-17.04). Dnipro: University of Customs and Finance.

ЗМІСТ

ВСТУП

Одним з найважливіших завершальних етапів навчального процесу є написання кваліфікаційної роботи бакалавра. Він є важливим показником знань, навичок і компетентностей студента, набутих за весь період навчання. Даний етап дозволяє всебічно оцінити рівень професійної підготовки випускника, його вміння застосовувати теоретичні знання на практиці та вирішувати конкретні професійні завдання.

Сучасний світ, особливо з 2019 року, почав переходити в онлайн простір, через що попит на дистанційне навчання значно зріс. Онлайн-освіта є однією з галузей, що досить стрімко розвивається: все більше викладачів переходять у цифровий формат, створюють власні курси та відкривають свої школи. Все більше людей віддають перевагу онлайн курсам завдяки їхній доступності та зручності. Статистика показує, що глобальний ринок онлайн-освіти зріс з $187 мільярдів у 2019 році до більш ніж $250 мільярдів у 2022 році, з прогнозом досягнення $470 мільярдів до 2026 року [1].

Найпоширенішим напрямком в онлайн освіті є вивчення іноземних мов. Це надає можливість подорожувати, знайомитись з культурами інших країн, спілкуватись з носіями мови, працевлаштовуватись закордоном або навчатись в іноземних університетах. Вивчення іноземних мов онлайн стало набагато зручніше завдяки різноманітності інформаційних технологій. Таку популярність цього напрямку зумовлено кількома факторами:

1. Глобалізація економіки та ринку праці потребує від фахівців знання мінімум однієї іноземної мови.

2. Розширення можливостей для міжнародної співпраці, подорожей та культурного обміну.

3. Зростання кількості міжнародних компаній, що потребують багатомовних співробітників.

4. Можливість працевлаштування закордоном або навчання в іноземних університетах.

5. Розвиток інформаційних технологій, які значно полегшують процес вивчення іноземних мов.

Не зважаючи на високий попит на вивчення іноземних мов, значна частина шкіл та приватних викладачів продовжують використовувати застарілі методи залучення нових клієнтів, обмежуючись лише оголошеннями на OLX та соціальними мережами. Отже, відсутність єдиного рішення, яке б поєднувало всі необхідні функції, значно ускладнює організацію навчального процесу та залучення учнів.

Об'єктом дослідження є сучасні технології проєктування та розробки вебсайту.

Предмет дослідження – проєктування та розробка сайту школи іноземних мов.

Метою даної кваліфікаційної роботи бакалавра є проєктування та розробка функціонального, зручного та ефективного вебсайту для школи іноземних мов.

Основна ідея проєкту полягає у створенні єдиної онлайн-платформи, яка дозволить потенційним учням ознайомитись зі школою та її послугами, обрати відповідний курс чи програму навчання, здійснити оплату в режимі онлайн, а також отримати доступ до додаткових матеріалів і ресурсів.

Така робота вимагає багато часу та виконання конкретних завдань. До основних належать:

1. Здійснити аналіз сучасних вебтехнологій та визначити найкращий стек для розробки платформи, враховуючи вимоги до функціоналу, ефективності та захисту даних.

2. Розробити оптимальну структуру сайту, яка забезпечить інтуїтивно зрозумілу навігацію для користувачів різних категорій. Це передбачає створення логічної ієрархії сторінок, розробку зручного меню та створення продуманої системи внутрішніх посилань для органічного переміщення користувача на сайті.

3. Створити інтерфейс з адаптивним дизайном, який буде ефективно працювати на різних пристроях, незалежно від розміру екрана. Він повинен забезпечувати однаково якісний досвід на смартфонах, планшетах та настільних комп'ютерах. Це вимагає використання сучасних технологій верстки з застосуванням фреймворків для адаптивного дизайну.

4. Розробити та інтегрувати систему онлайн-оплати, яка забезпечить зручний, швидкий та безпечний спосіб придбання курсів. Це передбачає інтеграцію надійних платіжних шлюзів, розробку зручного інтерфейсу для здійснення транзакцій та впровадження заходів захисту особистих даних, відповідно до сучасних стандартів безпеки.

5. Створити функціональний блог та інтерактивні елементи, які дозволять підвищити залученість користувачів, сприятимуть довшому перебуванню на сайті та збільшенню конверсії.

6. Провести тестування створеного вебсайту на різноманітних пристроях та в браузерах для забезпечення безперебійної роботи в усіх поширених програмних середовищах.

Вивчення функціоналу існуючих сайтів шкіл іноземних мов є досить важливим етапом аналізу. Зазвичай такі сайти пропонують базовий набір функцій: опис послуг, курсів та методик навчання, інформацію про викладачів, відгуки студентів та контактні дані. Однак, часто ці рішення мають обмежену функціональність, застарілий дизайн або відсутність важливих елементів, як:

1. Повноцінна інтеграція системи онлайн-оплати;
2. Адаптивний дизайн для мобільних пристроїв;
3. Інтерактивні елементи для залучення користувачів;
4. Регулярно оновлюваний контент у вигляді блогу чи новин;
5. Оптимізація для пошукових систем.

Також, крім того, багато існуючих рішень було створено за допомогою шаблонних конструкторів сайтів, що обмежує можливість кастомізації та розширення функціоналу відповідно до специфічних потреб школи.

Таким чином, існує значний попит на індивідуальні рішення, які б поєднували доступність, функціональність та можливість подальшого розвитку відповідно до зростання бізнесу. Завершення проєкту принесе суттєву користь для онлайн-школи у декількох аспектах:

1. Наявність професійно розробленого вебсайту допоможе значно розширити базу користувачів, які прагнуть вивчати іноземні мови та забезпечити доступ до послуг школи 24/7 з будь-якої точки світу.

2. Впровадження системи онлайн-оплати послуг значно полегшить процес адміністрування школи та навчання, зменшить навантаження на персонал та мінімізує можливість помилок при обробці замовлень.

3. Інтуїтивно зрозумілий інтерфейс та адаптивний дизайн сайту забезпечать зручне користування вебресурсом на будь-яких пристроях. Це сприятиме збільшенню задоволеності від користування сайтом.

4. Завдяки впровадженню системи аналітики, школа отримає корисні дані про поведінку користувачів та їх вподобання, що дозволить оптимізувати контент та маркетингові стратегії.

У перспективі подальшого розвитку та масштабуванні діяльності школи, платформу також чекають зміни: будуть додаватись нові навчальні програми, оновлюватиметься дизайн, розширюватимуться функціональні можливості. Крім того, перспективним напрямком розвитку буде створення особистого кабінету для учнів, який зміг би об'єднати всі аспекти навчального процесу на одній платформі, включаючи доступ до навчальних матеріалів, відстежування успішності та спілкування.

Таким чином, розробка вебсайту для школи іноземних мов є не просто актуальним, а й необхідним рішенням, яке допоможе обійти багатьох конкурентів на ринку праці.

Структура та обсяг кваліфікаційної роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. Роботу подано на 63 сторінках, в роботі міститься 24 рисунки, 5 таблиць та 5 додатків. Список використаних джерел складає 36 найменувань.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області, літературних джерел та постановка задачі

Сфера онлайн-освіти займає велику частку повсякденного життя людей і стрімко розвивається. Особливо попит на онлайн-освіту дуже зріс в період Covid-19, коли необхідно було сидіти вдома та знайти чим зайнятись. Після закінчення локдаунів багато людей почали знову подорожувати за кордон, що потребувало знання іноземних мов. Знання іноземних мов відкриває велику кількість можливостей для особистого та професійного розвитку, підвищує конкурентоспроможність на ринку праці та допомагає комунікувати з різними людьми.

Все більше викладачів та шкіл переходять в онлайн-простір, розуміючи як це зручно та прибутково, починають створювати свої продукти, курси, заняття та продавати в Інтернеті. В сучасному світі більшість людей надають перевагу онлайн-освіті, завдяки її зручності та доступності. Онлайн-школи вивчення іноземних мов дозволяють отримувати якісні послуги, незалежно від місця проживання, розробити гнучкий графік навчального процесу та використовувати інтерактивні матеріали для зручності.

Хоча, попри всі ці фактори, все ще дуже багато шкіл та викладачів, які не використовують можливості онлайн-простору або просто використовують застарілі методи просування. Наявність вебсайту для школи іноземних мов є невід'ємним пунктом, коли йде мова про залучення клієнтів та презентування себе в Інтернеті.

І хоча онлайн-навчання має велику низку переваг, все ж таки існують деякі недоліки, такі як якість зворотного зв'язку, автоматизація процесів оцінювання, збору даних і аналізу прогресу, а також зручність та інтуїтивність користувацького інтерфейсу. Велика кількість існуючих платформ не задовольняють потреби користувачів через, наприклад,

недостатню інтерактивність або відсутність єдиного рішення, яке б поєднувало всі необхідні користувачам функції. Багато шкіл мають незручні та незрозумілі інтерфейси вебсайтів та часто відсутність детального опису школи, послуг та навіть викладачів є перешкодою для придбання послуг/продуктів онлайн.

Аналіз літературних джерел є не менш важливим етапом у підготовки та розробці проєкту сайту школи іноземних мов, оскільки це надає можливість вже мати відповіді на можливі питання в майбутньому.

1. Технічна література з веброзробки.

Книга Бена Фрейна "HTML5 and CSS3: Building Responsive Websites" є дуже цінним ресурсом, щоб навчитись розробляти гарні адаптивні вебсайти. В книзі автор багато розповів як використовувати флексбокси та гріди для створення гнучких макетів. Також досить докладно було описано про застосування медіазапитів, щоб зробити дизайн адаптивним [2].

Крім того, для створення ігрових компонентів на вебсайті школи дуже допоможе книга Крістофера Пітта "Making Games with JavaScript". Це одна з найкращих знахідок у сфері розробки інтерактивних елементів на сайті [4].

2. UX/UI дизайн освітніх платформ.

Що стосується дизайну, то книга Стива Круга "Don't make me think, revisited: a common sense approach to web usability" просто скарб в цьому плані. В книзі описується важливість інтуїтивно зрозумілого дизайну та його основні принципи [5].

3. Гейміфікація в освіті.

Стосовно впровадження елементів гейміфікації в освітній процес, в книзі "Gamify your classroom: a field guide to game-based learning" пропонуються практичні підходи. Це допоможе зрозуміти що і як треба застосувати при розробці інтерактивних елементів на сайті, щоб підвищити мотивацію та залученість учнів [6].

Таким чином, аналіз цих джерел надає гарну основу для розробки сайту школи іноземних мов, що поєднує в собі ефективні педагогічні підходи,

сучасні вебтехнології та інтуїтивний дизайн. Це надає можливість створити не просто інформаційний ресурс, а повноцінну платформу, яка відповідає сучасним стандартам та потребам користувачів.

Отже, спираючись на вище приведені фактори, метою кваліфікаційної роботи є проєктування та розробка функціонального та зручного вебсайту для онлайн школи з вивчення іноземних мов, який забезпечить єдине рішення для користувачів та відповідатиме їх вимогам та потребам.

Основним завданням є надання можливості потенціальним учням ознайомитись зі школою та послугами, протестувати свої знання з різних тем, придбати продукт пропонований онлайн-школою та отримання нових знань зі статей в блозі.

Отже, для досягнення поставленої мети необхідно виконати низку завдань, як:

1. Провести аналіз вже існуючих рішень для онлайн-платформ із вивчення іноземних мов для виявлення їх сильних та слабких сторін.

2. Визначити та описати основні функціональні та нефункціональні вимоги до вебсайту онлайн-школи.

3. Розробити структуру сайту та дизайн користувацького інтерфейсу відповідно до цільової аудиторії та потреб користувача.

4. Реалізувати основні модулі сайту, як інформаційні сторінки, каталог курсів, кошик, реєстрацію, онлайн-оплату, мовне перемикання, освітні ігри та адаптивну верстку.

5. Провести тестування функціональності та зручності користування для надання оцінки ефективності створеного рішення.

6. Підготувати систему для експлуатації та можливого подальшого розширення, як створення особистого кабінету.

Для реалізації поставлених завдань та досягнення мети кваліфікаційної роботи, було використано наступні методи:

1. Проведено аналіз літературних джерел та існуючих рішень. З самого початку було проаналізовано актуальність створення вебзастосунків в даній

сфері, вивчено принципи UX/UI-дизайну та функціональність подібних онлайн-платформ з вивчення іноземних мов.

2. Моделювання структури вебсайту за допомогою Figma. На основі попереднього аналізу було зроблено моделювання структури онлайн-школи, розроблено прототипи інтерфейсу на Figma та визначено основну функціональність вебсайту.

3. Розробка програмного забезпечення. Цей метод містить в собі реалізацію вебсайту за використанням сучасних технологій веброзробки. Особливу увагу було приділено вибору технологій, фреймворків та забезпечення адаптивності й зручності використання.

4. Тестування та оцінка ефективності. Після етапу розробки було проведено функціональне тестування системи із залученням реальних користувачів. Для оцінки ефективності використання платформи було застосовано методи юзабіліті-тестування та аналіз результатів навчання.

Застосування такого комплексного підходу допомагає врахувати сучасний стан предметної області та також створити єдине рішення, яке відповідає основним потребам користувачів. Це допомагає краще зрозуміти болі користувачів та їх потреби взагалі, оптимізувати структуру платформи, продумати всі основні моменти заздалегідь та забезпечити якість та ефективність розробленого програмного забезпечення. Реалізація даного проєкту принесе багато користі, не тільки онлайн-школі, а й самому ринку онлайн-освіти в цілому.

Буде створено функціональну онлайн платформу, яка дозволить учням ознайомитись зі школою та вчителями, реєструватись та обирати курси, тестувати свої знання з різних тем та різних мов у вигляді інтерактивних ігор. Також завдяки всім інтерактивним елементам та гейміфікації очікується підвищення мотивації та залученості потенційних учнів. Обраний спосіб реалізації також надає можливість масштабування платформи при подальшому розвитку онлайн-школи, наприклад як додавання нових мов і курсів, особистого кабінету та різноманітного функціоналу. Реалізована

система забезпечить підвищення довіри серед користувачів та сприятиме розширенню доступу до якісної мовної освіти широкому колу користувачів, незалежно від їхнього місця проживання.

1.2 Аналіз користувачів, бізнес-моделі та конкурентного середовища

Перш ніж переходити до безпосередньої розробки вебсайту, необхідно чітко визначити цільову аудиторію майбутнього проєкту, її цілі та очікування. Загалом цільова аудиторія цього вебсайту – це потенційні користувачі віком 14+, незалежно від статі та місця проживання, які прагнуть вивчати іноземні мови та хочуть знайти місце, де всі необхідні знання структуровано зібрані у єдину купу.

Основна аудиторія вебсайту школи іноземних мов може бути представлена трьома ключовими сегментами, кожен з яких має власні характеристики та потреби.

1. Підлітки 14-18 років – школярі, які зацікавлені у покращенні знань з іноземних мов для підготовки до іспитів як ЗНО та НМТ, або для отримання додаткової мовної практики. Також значну роль відіграють соціальні фактори, як спілкування з однолітками онлайн, розуміння пісень, фільмів та відеоігор мовою оригіналу. Характерною особливістю даного сегменту є потреба в яскравому, візуально привабливому інтерфейсі з елементами гейміфікації для підтримання інтересу та мотивації до навчання.

2. Студенти 18-23 років – люди, які зацікавлені у вивченні мов для досягнення більш масштабних життєвих цілей. Для них це засіб для міжнародної академічної мобільності, подорожей, пошуку роботи та складання міжнародних мовних іспитів. Для цієї групи користувачів дуже важлива якість та актуальність навчальних матеріалів, а також можливість навчатись в зручний для них час.

3. Дорослі 24+ років – люди, яким іноземна мова потрібна для професійної кар'єри, переїзду за кордон та в рамках саморозвитку. Ця

категорія найбільш різноманітна за своїми потребами та характеристиками. Дорослі користувачі надають перевагу чіткій структурі вебсайту з інтуїтивно зрозумілою навігацією та системою оплати.

Цифрова грамотність користувачів є досить важливим аспектом для аналізу, оскільки вони можуть дуже різнитись – від початківців, яким важлива простота та інтуїтивність інтерфейсу, до просунутих, які очікують широкого функціоналу та можливостей персоналізації. Отже, сайт має бути адаптований на різні рівні технічної підготовки, пропонуючи базові функції, зрозумілі всім, і розширені можливості для більш досвідчених користувачів. Важливо передбачити інструкції та розділ з відповідями на поширені запитання.

Також необхідно передбачити технічні вимоги до користувачів, тобто пристроїв, з яких вони здійснюють доступ до вебсайту. Сучасна аудиторія використовує різноманітні пристрої – від мобільних телефонів до стаціонарних комп'ютерів. Саме тому сайт повинен бути адаптивним та забезпечувати комфортну роботу на екранах різного розміру. При цьому треба врахувати особливості кожного типу пристроїв, наприклад, для мобільних телефонів важлива компактність, для планшетів та ноутбуків – зручність навігації та доступність функцій.

Таким чином, детальний аналіз користувачів вебсайту школи іноземних мов включає вивчення потреб і очікувань цільової аудиторії, технічних можливостей і обмежень. Такий комплексний підхід дозволяє створити вебсайт, який максимально відповідатиме потребам різних сегментів користувачів.

У ході кваліфікаційної роботи розробляється вебсайт для онлайн-вивчення іноземних мов. Основною метою якого є надання можливості користувачам придбати структуровані курси та проходити їх у зручному темпі, використовуючи цікаві та якісні навчальні матеріали. Для формалізації цих цілей та визначення ключових функціональних блоків системи було використано Business Model Canvas.

У контексті розробки вебсайту, бізнес-модель слід розглядати не тільки як фінансову структуру, але і як технічну екосистему, яка забезпечує функціонування освітньої платформи та реалізацію її основних завдань. Після аналізу потреб різних сегментів користувачів, можна визначити ключові технічні складові бізнес-моделі. Ця модель допоможе зрозуміти які саме функціональні та нефункціональні вимоги треба реалізувати у системі для підтримки бізнес-процесів школи.

Важливо зазначити, що структура доходів та структура витрат з бізнес-моделі безпосередньо впливають на архітектурні рішення та вибір технологій розробки. Наприклад, бізнес, заснований на платній підписці, буде потребувати впровадження системи авторизації, розрахунків та управління контентом з гнучкими правами доступу.

Головні напрямки діяльності та стратегічні партнери впливають на потребу в конкретних функціях, як інтеграція з соціальними мережами, платіжними сервісами або розробка програмних інтерфейсів для партнерських ресурсів. Створену бізнес-модель для розробленого проєкту наведено на рисунку 1.1.

Рисунок 1.1 – Бізнес-модель Business Model Canvas

Використання Business Model Canvas на етапі проєктування дозволяє інтегрувати бізнес-підхід у процес інженерії програмного забезпечення. Сегментація клієнтів допоможе визначити вимоги до користувацьких ролей та сценаріїв взаємодії. Ціннісна пропозиція грає велику роль у визначенні функціоналу платформи: що повинно на ній бути та як вона має функціонувати. Канали впливають на вибір технологій розробки вебінтерфейсу, мобільної адаптації та інтеграції з додатковими системами.

У сучасному digital-просторі України вивчення іноземних мов все більше переходить в онлайн-формат. Саме це створює конкурентне середовище з різноманітними підходами до навчання. Такий аналіз дозволить виявити ключові характеристики успішних освітніх платформ та визначити потенційні напрямки для вдосконалення розробляємого проєкту.

Далі було проаналізовано чотири популярні школи на ринку онлайн-освіти, для розуміння поточних тенденцій та особливостей користувацького досвіду. Більш детальна інформація щодо аналізу надана в таблиці 1.1.

Таблиця 1.1

Аналіз існуючих рішень на ринку онлайн освіти в Україні

Критерій	Englishdom	SpeakNow	Educado	NewBrain
Дизайн сайту	Нестандартний, без звичайного меню для десктопу, вимагає скролінгу для навігації	Традиційний, дещо застарілий	Сучасний, нестандартний, креативний	Інтуїтивний дизайн, сучасний
Інформація про викладачів та школу	Детальні профілі з фото, досвідом, відеопрезентацією	Обмежена інформація, базові дані	Креативні профілі, але обмежена інформація	Відеопрезентація, обмежена інформація
Додаткові матеріали	Мобільний додаток, відеоконтент	Обмежений набір матеріалів	Немає	Немає
Зручність інтерфейсу	Потребує звикання	Складна навігація	Інтуїтивно зрозуміла	Інтуїтивно зрозуміла
Наявність мобільної версії	Повноцінний додаток	Адаптивний сайт	Адаптивний сайт	Адаптивний сайт

Серед популярних онлайн-шкіл в Україні можна визначити Englishdom, SpeakNow, Educado та NewBrain. Кожна з них має свої особливості дизайну та подачі інформації.

Englishdom є найпопулярнішою школою на ринку, має нестандартний дизайн сайту, але відсутність звичайного меню змушує користувача довго шукати інформацію. SpeakNow використовує консервативний дизайн, що може не подобатись молодці. Також на сайті бракує інформації про викладачів, а навігація потребує покращення. Сайт Educado відрізняється яскравим сучасним дизайном з елементами гейміфікації, але сайту бракує детальної інформації про викладачів та курси. NewBrain має інтуїтивний сайт з гарним описом послуг, однак також трохи бракує опису викладачів.

Таким чином, проведений огляд існуючих рішень на ринку онлайн-освіти іноземних мов є важливим етапом для успішного проєктування вебсайту. Також, знаючи досвід користувачів на сайтах конкурентів, можна створити інтерфейс, який враховує найкращі практики. Розуміючи те, які функції є стандартом ринку, а які можуть стати конкурентною перевагою, допоможе правильно проєктувати функціонал.

1.3 Вимоги до системи, структуризація функціоналу та сценарії взаємодії

Перед тим, як переходити безпосередньо до проєктування та реалізації системи, необхідно чітко визначити та сформулювати вимоги до неї. Вимоги, в свою чергу, формуються на основі очікувань цільової аудиторії, завдань, які має виконувати система та обмежень. Вимоги зазвичай поділяються на функціональні та нефункціональні.

Функціональні вимоги визначають поведінку системи, описують функції та можливості, які повинна забезпечувати система, для того, щоб задовольнити потреби користувачів та досягти бізнес-цілей. До таких належать наступні.

1. Реєстрація та авторизація користувачів. Система повинна надавати можливість користувачам створювати обліковий запис із зазначенням електронної пошти та паролю. Система повинна перевіряти відповідність пароля вимогам безпеки, як стосовно довжини пароля та наявності різних типів символів. Авторизація користувачів також здійснюється за допомогою електронної пошти та паролю, шляхом перевірки наявності та відповідності даних у базі даних.

2. Огляд та вибір курсів. Система повинна надавати користувачам структурований список усіх доступних курсів з можливістю перегляду більш детальної інформації. Кожен курс повинен мати свою особисту сторінку з повним описом програми, інформацією про тривалість, вартістю та результатами навчання.

3. Купівля курсів. Система має забезпечувати повний цикл процесу купівлі курсів. Повинна бути можливість додавання обраних курсів до кошика, перегляду його вмісту, зміни кількості обраних курсів та їх видалення. Для безпечного проведення онлайн-платежів з різними способами оплати, повинна бути інтеграція з платіжними системами.

4. Інформаційні сторінки. Система повинна передбачати наявність інформаційних сторінок, що надають необхідні користувачу відомості, як інформація про школу в цілому, інформація про викладацький склад та їх досвід, відгуки попередніх студентів, сторінка з відповідями на найпоширеніші запитання та контакти школи.

5. Блог. Система повинна мати блог для публікації цікавих навчальних матеріалів або новин школи. Блог повинен містити можливість перегляду всіх опублікованих статей з поділом за мовами.

6. Ігри. Системі потрібно мати інтерактивні елементи у виді ігор для вивчення або повторення якихось тем та слів з іноземних мов. Доступ до ігор можна отримати навіть без авторизації. Елементи повинні мати зручний інтерфейс та доступною для різних пристроїв.

7. Зміна мови інтерфейсу сайту. Система повинна підтримувати багатомовний інтерфейс для розширення аудиторії. Тобто, буде забезпечено повну локалізацію інтерфейсу українською та англійською мовами. Система буде мати зручний механізм для перемикання між мовами в шапці сайту.

Нефункціональні вимоги містять в собі якісні характеристики системи та обмеження. Вони не залежать від конкретної функціональності, але впливають на якість, продуктивність та зручність. Ці вимоги визначають основні аспекти користувацького досвіду, надійності та безпеки системи, що є найважливішим для загального успіху вебплатформи. Основними такими вимогами є:

1. Вимоги до архітектури системи. Архітектура системи повинна відповідати сучасним стандартам розробки вебдодатків. Клієнтська частина повинна використовувати HTML5, CSS3 та JavaScript в поєднанні з React-бібліотекою для покращення користувацького досвіду. Серверна частина має бути реалізована з використанням Firebase для простоти інтеграції та відповідності сучасним стандартам. Також для зберігання даних про користувачів, система повинна використовувати нереляційну базу даних Firebase Firestore.

2. Параметри продуктивності. Система повинна забезпечувати високу продуктивність для комфортної роботи користувачів. Це означає, що час завантаження сторінок повинен бути мінімальним, до 2 секунд при стабільному інтернет-з'єднанні. Повинна бути середня пропускна здатність – стабільне обслуговування не менше 50 одночасних користувачів без втрати продуктивності сайту. Також для швидкого завантаження сторінок необхідно використовувати методи оптимізації, як мінімізація CSS та JavaScript, оптимізація зображень.

3. Надійність та доступність. Система повинна забезпечувати коректну обробку помилок та виняткових ситуацій без порушення загальної роботи сайту. Також для того, щоб забезпечити можливість відновлення у разі збоїв,

необхідно впровадити регулярне автоматичне резервне копіювання даних бази даних.

4. Зручність використання. Система повинна мати високий рівень зручності користування для різних категорій користувачів. Основним є наявність інтуїтивного інтерфейсу, логічною структурою сайту та зрозумілою навігацією, що дозволить користувачам легко знаходити необхідну інформацію. Система повинна мати адаптивний дизайн для коректного відображення на різних пристроях. Також важливо мати забезпечення коректної роботи в усіх сучасних браузерах, як Chrome, FireFox, Safari та Edge.

5. Безпека. Для захисту персональних даних та автентифікації користувачів треба використати сервіс Firebase Authentication, який забезпечує надійне хешування паролів та безпечне управління сесіями. Передача всіх даних між клієнтом та сервером здійснюється за допомогою захищеного протоколу HTTPS, що гарантується платформою Firebase Hosting. Безпека фінансових транзакцій забезпечується інтеграцією з сертифікованою платіжною системою LiqPay, яка обробляє платіжні дані на своїй стороні відповідно до стандартів PCI DSS. Додатково, для контролю доступу до даних у базі Firestore застосовано правила безпеки.

Отже, представлені вимоги до системи є основою для початку проєктування та розробки вебсайту онлайн-школи іноземних мов. Завданням сайту буде забезпечення зручного доступу до інформації про курси, можливість їх придбання та ефективну взаємодію клієнтами зі школою.

Для опису функціональності вебсайту та відображення доступного функціоналу для кожної групи користувачів було створено функціональну модель системи. Модель, яку було реалізовано діаграмою варіантів використання (Use Case Diagram), відображає основні сценарії взаємодії користувачів із системою та взаємозв'язки між різними ролями й діями в межах вебсайту школи іноземних мов (рисунок 1.2).

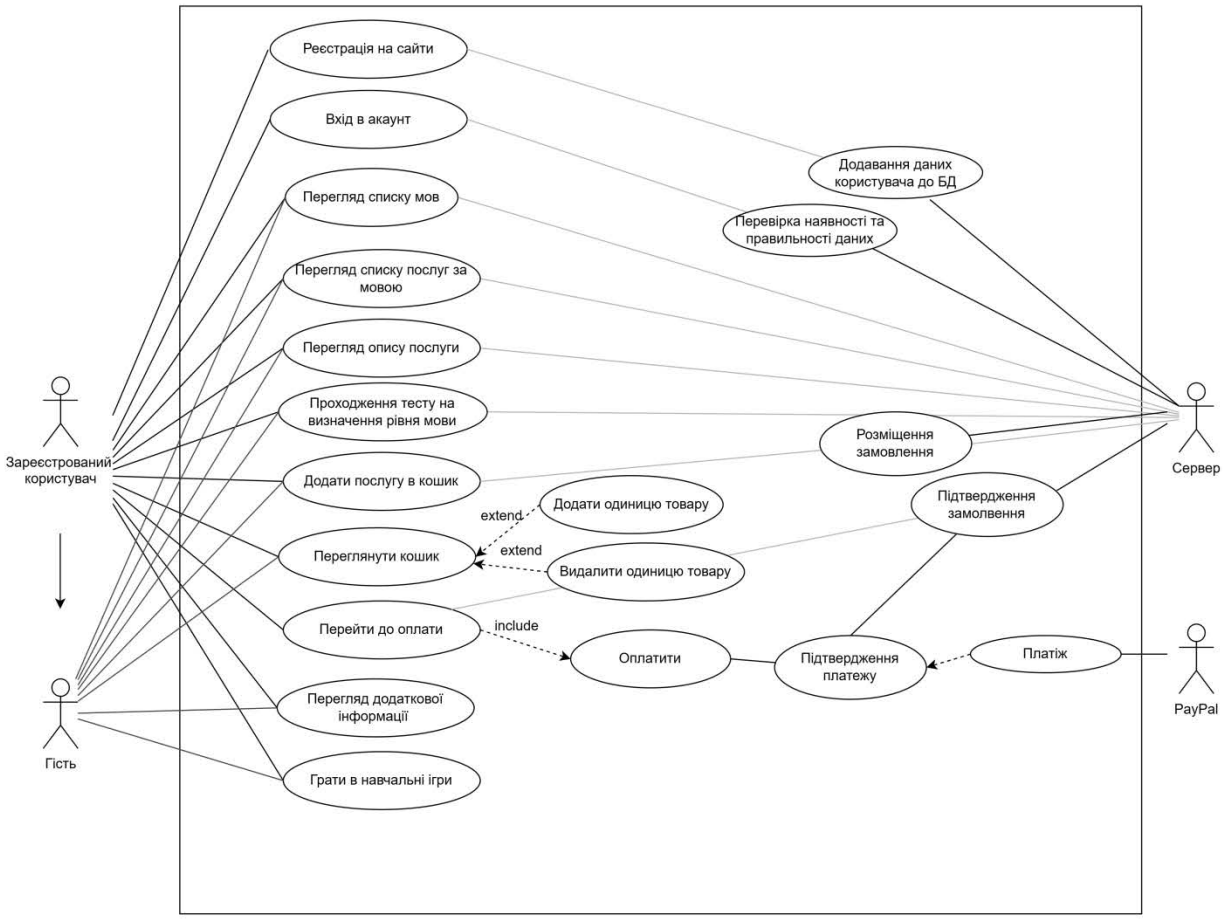


Рисунок 1.2 – Діаграма сценаріїв взаємодії

На діаграмі сценаріїв взаємодії зображено 4 основні актори:

1. Гість – користувач, який не пройшов авторизацію;

2. Зареєстрований користувач – користувач, який увійшов у систему;

3. Сервер – бекенд частина, яка відповідає за обробку даних;

4. Платіжна система – зовнішній сервіс для здійснення онлайн-оплати.

Наведена діаграма демонструє всі сценарії взаємодії користувача з вебсистемою школи іноземних мов, відображаючи взаємозв'язки між різними учасниками системи.

Така діаграма варіантів використання надає змогу чітко визначити всі можливі сценарії взаємодії користувача із системою, а також взаємозв'язки між функціями, серверною частиною та іншими зовнішніми сервісами. Це дозволяє досить ефективно розробити архітектуру та подальшу реалізацію сайту, урахувавши потреби користувачів та технічні вимоги.

1.4 Висновки до першого розділу

Отже, проведене глибоке дослідження предметної області та аналіз вимог до розробки вебплатформи для школи іноземних мов дозволили сформувати розуміння контексту проєкту та визначити ключові напрямки подальшої роботи.

У процесі дослідження предметної області, було встановлено, що сегмент онлайн-освіти у сфері вивчення іноземних мов досить швидко зростає та має значний потенціал для подальшого розвитку. Через зростання важливості та актуальності іноземних мов, збільшення кількості виникнення мовних шкіл і викладачів на ринку праці, формується стійкий попит на якісні та доступні освітні послуги в цій галузі. Особливо актуальним таке рішення становиться в сучасних реаліях, і це пов'язано із дистанційним форматом взаємодії та необхідністю забезпечення безперервності освітнього процесу.

На основі проведеного аналізу було чітко окреслено основні цілі та завдання проєкту. А саме – створення комплексного вебрішення для школи іноземних мов з урахуванням сучасних тенденцій у веброзробці та освітніх технологіях. Платформа повинна забезпечувати повний цикл взаємодії користувача із системою: від реєстрації та вибору курсу для проходження навчання, до тестування набутих знань та здійснення оплати послуг. Також велику увагу було приділено інтерактивності та адаптивності вебсайту, що дозволить забезпечити цікавий підхід до навчання та підвищить його ефективність.

Детальне вивчення профілю цільової аудиторії дозволило виділити основні сегменти потенційних користувачів та визначити їх потреби та очікування щодо вебсайту. Зокрема, було враховано особливості сприйняття та взаємодії з системою користувачів різних вікових категорій та цифровою грамотністю. Все це дало можливість сформувати перелік вимог до інтерфейсу, функціоналу та UX-дизайну, що забезпечить максимальну зручність у користуванні системи для кожної з цільових груп.

Також в рамках дослідження було розроблено комплексну бізнес-модель школи іноземних мов, яка включає в себе структуру джерел доходу, систему ролей і повноважень користувачів, а також модель основних бізнес-процесів. Це дозволило зрозуміти чи відповідають технічні рішення бізнес-цілям проєкту та створити передумови для майбутньої масштабованості системи відповідно до зростання бізнесу.

Одним із важливих етапів дослідження став порівняльний аналіз конкурентних рішень. В ході аналізу було вивчено функціональні можливості, інтерфейси та бізнес-моделі провідних платформ в Україні для вивчення іноземних мов. В процесі було виявлено слабкі та сильні сторони вебрішень і було виявлено які потреби користувачів задовольняються, а які залишаються незакритими. На основі цього аналізу було визначено унікальні переваги, яка повинна мати розроблювана система, для того, щоб забезпечити конкурентоспроможність на ринку праці.

Таким чином за результатами проведених досліджень, можна було сформулювати чіткий переліг функціональних вимог, які характеризують очікувану поведінку системи для задоволення потреб користувача. А також нефункціональних вимог, які визначають якісні характеристики системи – безпеку, надійність, продуктивність, масштабованість тощо.

Отже, в ході дослідження було обґрунтовано вибір підходів до проєктування та розробки системи, урахувавши специфіку предметної області та поставлених завдань. Після аналізу переваг та недоліків різноманітних методологій, було обрано оптимальні підходи, які допоможуть забезпечити ефективну організацію процесу розробки, контроль якості фінального продукту та відповідність отриманих результатів вимогам замовника і потенційних користувачів.

Таким чином, перший розділ роботи заклав міцний теоретико-методологічний фундамент для подальшого проєктування та розробки вебплатформи для школи іноземних мов. Він окреслює основні напрями роботи та забезпечує цілісне розуміння контексту проєкту.

РОЗДІЛ 2. ПРОЄКТУВАННЯ АРХІТЕКТУРИ САЙТУ

2.1 Вибір архітектурних рішень, технологій та середовища розгортання

Кожне програмне забезпечення має свій життєвий цикл, тобто час від створення будь-якого програмного продукту до кінця його розробки та впровадження. Через що його можна представити у вигляді якоїсь моделі. Вибір моделі життєвого циклу програмного забезпечення є дуже важливим етапом при плануванні проєкту. До основних завдань цих моделей належать:

1. Структуризація та організація процесу розробки. Модель життєвого циклу допомагає визначити послідовність етапів виконання для створення якісного програмного продукту. Допомагає розробити чітко визначену дорожню варту з етапами, завданнями та результатами. Також є гарним засобом для розподілу людських, часових та фінансових ресурсів.

2. Управління ризиками. Різні моделі допомагають знайти різні підходи для виявлення потенційних проблем на ранніх стадіях та мінімізувати ймовірність непередбачених ситуацій.

3. Керування якістю продукту. Це допомагає у визначенні часу та виду проведення тестування для забезпечення якості програмного продукту. Також різні моделі мають різні вимоги до документації, коду та перевірок. Модель життєвого цикла допомагає забезпечити процеси перевірки відповідності продукту вимогам та потребам користувачів.

4. Адаптація до специфіки проєкту. Різні типи проєктів потребують суттєво різних підходів для організації процесу розвитку. За допомогою конкретного контексту проєкту дозволяє вибрати найкращі практики, які відповідатимуть реальним вимогам та обмеженням. Важливим також є те, що деякі моделі краще підходять для малих проєктів, а інші – для великих.

Таким чином, після ретельного аналізу було вирішено обрати ітеративну модель розробки життєвого циклу програмного забезпечення для даного кваліфікаційного проєкту. Ітеративна модель дуже добре відповідає

специфіці розробки вебсайту для школи іноземних мов. Вона дозволяє розбити проєкт на логічні модулі та впроваджувати їх поетапно, тобто зручно поступово розширювати функціональність.

До основних переваг цієї моделі в контексті розробки вебсайту для школи іноземних мов належать:

1. Швидке виведення базової версії сайту. Оскільки вже після першої ітерації можна отримати базовий функціональний вебресурс, це дає змогу та час зосередитись на більш важливих на складних компонентах. Вже на цьому етапі можна зібрати зворотний зв'язок від зацікавлених сторін та своєчасно внести необхідні зміни.

2. Поетапне впровадження функціональних можливостей. Тобто, спочатку реалізується базова інформаційна архітектура із ключовими розділами, а вже потім можна переходити до більш складних елементів, наприклад як багатомовний інтерфейс, інтерактивних компонентів та баз даних. Таким чином можна раціонально розподілити ресурси та зосередитись на пріоритетних завданнях.

3. Висока адаптивність до змін. Зазвичай на початковому етапі може бути досить складно чітко сформулювати всі вимоги до проєкту, але ця методологія дозволяє гнучко реагувати на уточнення вимог. Наприклад, на будь-якому етапі буде можливість впровадити систему онлайн-реєстрації чи деяку модифікацію структурних елементів сайту.

Отже, аналізуючи всі переваги, в межах даного проєкту ітеративну модель було обрано як модель життєвого циклу. А що стосовно методології розробки, то було обрано Agile-підхід. Оскільки ця методологія добре доповнює ітеративну модель, оскільки вона є гнучкою та адаптивною. Її головна мета повністю відповідає меті даної кваліфікаційної роботи – створення цінного програмного продукту, який відповідає очікуванням користувачів та може оперативно адаптуватись до змін вимог.

Після аналізу моделей та методологій розробки програмного забезпечення, важливим етапом є правильний вибір технологічного стека.

При цьому необхідно враховувати сучасні тенденції у веброзробці, вимоги до системи та зручність підтримки майбутнього продукту.

Отже, що стосується мов програмування і технологій, то вибір був досить простим. Верстання вебсайту буде здійснено за допомогою мови гіпертекстової розмітки HTML5, який є основою сучасної веброзробки. Він дозволяє додати сайту семантичну структуру за допомогою семантичних елементів для покращення SEO та має розширені мультимедійні можливості. HTML5 є поточним вебстандартом та гарантує кросбраузерність та доступ контенту для широкого кола споживачів.

Що стосується візуального оформлення та стилізації, то тут буде використовуватись CSS3. Він допомагає розробити сучасні, естетично привабливі та зручні інтерфейси для вебсайтів. Для розробки адаптивного дизайну, було вирішено використовувати чистий CSS із використанням медіазапитів та гнучких макетів. Таким чином досить просто створити адаптивний дизайн, який буде добре відображатись на різних пристроях, не залежачи від зовнішніх CSS-фреймворків.

За інтерактивність на сайті відповідає JavaScript, з його допомогою можна переміщатись між сторінками, читати блог та грати в ігри. Він дуже важливий якщо необхідно зробити обробку дій користувача на сайті для передачі якихось даних на сервер. JavaScript дозволяє створити інтерактивні елементи, такі як слайдери або випадаючі меню.

В якості основного фреймворку для фронтенду було обрано React. Він використовується для створення користувацьких інтерфейсів, шляхом розробки складних, масштабований односторінкових додатків на основі компонентного підходу. Це значно спрощує розробку та тестування проєкту. І що ще важливо, то React використовує Virtual DOM, тобто будуть оновлюватись лише ті частини сторінки, які зазнали змін. Такий компонентний підхід ідеально підходить для вебсайту школи, де є багато елементів з повторюваною структурою.

Далі необхідно було обрати інтегроване середовище розробки, і було вирішено використовувати Visual Studio Code (VS Code), оскільки воно підтримує основні використовувані технології та має функції інтелектуального підсвічування синтаксису. Також VS Code має вбудований термінал, що дуже спрощує роботу з командним рядком.

Для того, щоб ефективно управляти кодовою базою проєкту та відстежувати зміни було обрано розподілену систему контролю версій Git. Він дозволяє фіксувати кожну зміну в коді та зберігає це в історію. Саме це дає можливість повернутись до будь-якої версії коду за необхідності.

Щодо бібліотек, то було вирішено використовувати бібліотеку React Router для управління навігацією. Вона дозволяє організувати багаторівневу навігацію між різними розділами сайту без перезавантаження сторінки. Ця бібліотека підтримує динамічні маршрути, що робить її дуже зручним для відображення сторінок окремих курсів або навчальних матеріалів.

Для зберігання даних користувачів для входу та реєстрації на вебсайті школи іноземних мов було обрано нереляційну базу даних FireBase Firestore. Найзручнішим є те, що ця платформа звільняє від необхідності налаштування, розгортання та підтримки власного серверного застосунку для роботи з базою даних. Вибір такого типу бази даних було зумовлено тим, що навідміну вед реляційних баз даних, вони дозволяють зберігати документи з різною структурою в одній колекції.

Firebase це вже готова система аутентифікації, яка пропонує готові механізми для реєстрації, входу та навіть входу через Google та інші соціальні мережі. Ця хмарна платформа від Google може захистити дані та зберігати їх на своїх серверах. Для розробляємого вебсайту така платформа буде досить зручною на початкових етапам, а вже далі при масштабуванні школи та їх сайту можна буде замінити Firebase на іншу технологію.

Такий обраний технологічний стек є збалансованим рішенням, яке відповідає сучасним вимогам веброзробки та потребам школи та її користувачів. Отже, комплексний аналіз усіх цих аспектів надає можливості

для визначення оптимальних шляхів розв'язання завдання з проєктування та розробки сайту школи іноземних мов, забезпечуючи створення ефективної та зручної платформи.

Наступним важливим пунктом для успішного запуску та безперебійної роботи вебсайту буде врахування певних технічних вимог до середовища, де буде працювати розроблена система. Спочатку аналізуємо вимоги до сервера.

1. Мінімальні вимоги до апаратного забезпечення сервера. Для розробляємого сайту потрібно використовувати сервер із достатніми технічними характеристика, щоб все працювало коректно. До мінімальних вимог до апаратного забезпечення належить процесор, наприклад, Intel Core i5 або AMD Ryzen 5, з тактовою частотою в 2.5 GHz. Це дозволить швидко та без затримок обробляти запити користувачів, а також швидке завантаження всіх компонентів React. Для мінімальної роботи сайту, обсяг оперативної пам'яті має бути не менше 8 GB.

2. Вимоги до серверного програмного забезпечення. Оскільки весь проєкт базується на Firebase як серверному рішенні, то традиційне серверне програмне забезпечення не потрібне. Для розробки проєкту необхідно встановити Node.js для використання менеджера пакетів. Також середовище розробки повинно підтримувати JavaScript та JSX для розробки React-компонентів.

3. Вимоги до хостингу та необхідні серверні розширення і модулі. Для даного React-застосунку з Firebase краще використовувати Firebase Hosting як основне середовище розгортання. Так як воно вже оптимізовано для роботи з іншими сервісами Firebase та забезпечує високу безпеку і продуктивність. Firebase Hosting при розгортанні надає глобальну CDN мережу, протокол HTTPS з SSL-сертифікатом, що дуже важливо для швидкого завантаження React-додатку.

І не менш важливим пунктом, є виявлення та опис вимог до операційних систем.

1. Вимоги до операційних систем для користувачів сайту. Для звичайних користувачів сайту підтримується широкий спектр операційних систем. На десктопних комп'ютерах React-застосунок буде коректно працювати з Windows 8.1 та у новіших версіях. Для користувачів Apple краще використовувати macOS Mojave (10.14) та новіші версії. Оскільки вони надають всю необхідну підтримку для роботи з React та Firebase Authentification. Для користувачів Linux це простіше, вони мають доступ до функціоналу сайту через будь-який сучасний дистрибутив Linux.

2. Підтримувані операційні системи мобільних пристроїв. Мобільна версія вебсайту адаптована до сучасних операційних систем мобільних пристроїв. Для користувачів iOS, починаючи з версії 14.0, зокрема власники iPhone 8 і новіших моделей, можуть розраховувати на стабільну роботу сайту. Для користувачів Android з версією 9.0 та вище, буде коректне відображення React-компонентів та надійне функціонування сервісів Firebase. Отже, за аналізом усіх вищезазначених вимог, можна буде гарантувати стабільну роботу вебсайту на широкому спектрі пристроїв і платформ. Зручність у використанні для споживача гарантує оптимізація під актуальні операційні системи та сумісність з популярними браузерами.

2.2 Проєктування структури та моделі даних

Ретельно продумана архітектура та логічно організована модель даних є ключовим аспектом в ефективному функціонуванні будь-якого вебзастосунку, а особливо для сайту школи іноземних мов.

Архітектуру вебсайту було розроблено використовуючи сучасні вебтехнології та реалізовано на основі клієнт-серверної моделі. Архітектуру побудовано на принципах компонентно-орієнтованого підходу з використанням бібліотеки React для клієнтської частини, Firebase для хостингу та автентифікації, та інтеграції з платіжною системою. Всю логіку інтерфейсу було реалізовано на стороні браузера, що відповідає моделі Single

Page Application. Загальну схему взаємодії ключових компонентів системи наведено на рисунку 2.1.

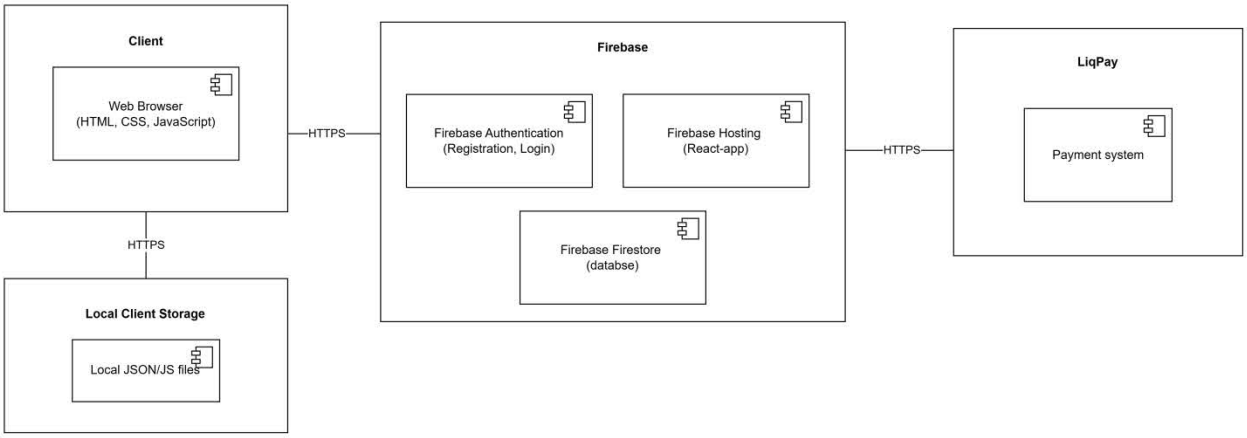


Рисунок 2.1 – Вигляд архітектури проєкту

Отже, розглянемо основні архітектурні блоки системи детальніше.

1. Клієнтська частина. Основним середовищем виконання клієнтської логіки є веббраузер. Тобто, користувач взаємодіє з вебсайтом через веббраузер, а він в той час відповідає за візуалізацію інтерфейсу, обробку дій користувача та забезпечує зв'язок з серверною частиною. Клієнтська частина сайту взаємодіє з платформою Firebase через захищене з'єднання HTTPS, що дозволяє безпечно проходити автентифікацію, отримувати та зберігати важливу інформацію, таку як дані про курси чи навчальні матеріали.

2. Локальне сховище клієнта. На стороні клієнта використовується локальне сховище браузера, де можуть зберігатись локальні JSON/JS файли. Це сховище використовується для зберігання тимчасової інформації, як обрана мова сайту, кошик тощо. Загалом, використання локального сховища зменшує кількість запитів до сервера та прискорює взаємодію користувача з інтерфейсом.

3. Платформа Firebase. Платформа Firebase має головну роль у серверній інфраструктурі та функціонує як Backend-as-a-Service. Тобто, частину типових серверних завдань як автентифікація та хостинг статичних

файлів платформа бере на себе. На рисунку виділено два ключові сервіси Firebase, як Firebase Authentification та Firebase Hosting.

4. Платіжна система LiqPay. Щоб користувачі мали можливість онлайн-оплати навчальних курсів, до архітектури сайту було інтегровано українську платіжну систему LiqPay. LiqPay діє як платіжний шлюз та дозволяє користувачам безпечно здійснювати онлайн-оплату, використовуючи різноманітні методи. Процес оплати досить стандартний. Клієнтська частина надсилає запит до серверної частини з параметрами платежу. Далі користувача перенаправляє на захищену платіжну сторінку LiqPay, де він вводить свої платіжні дані і LiqPay обробляє транзакцію.

Отже, перейдемо більш до логічної структури та моделі даних. Як вже було зазначено раніше, то для ефективного функціонування сайту школи іноземних мов було розроблено логічну модель даних. А вся інформація про користувачів, курси, викладачів буде зберігатись з бази даних Firestore. Сервер Firebase Authentification управляє даними користувачів, які потрібні для реєстрування та входу до системи. Кожен з зареєстрованих користувачів має запис в цій системі, та має унікальний ідентифікатор (uid).

Оскільки в цій роботі використовується нереляційна база даних, то структура таблиць трохи інша. В нереляційній базі даних є колекції, вони як контейнери для даних. Всередині кожної колекції є документи, які мають свої унікальні ідентифікатори. А вже всередені кожного документа є поля пари “ключ-значення”, де значенями можуть бути строки, числа тощо.

Розглянемо детально кожну колекцію.

1. Колекція users. Ця колекція зберігає інформацію про вже зареєстрованих користувачів системи. Вона обов’язково має унікальний ідентифікатор користувача, який зазвичай співпадає з UID з Firebase Authentification. Ця колекція має тільки необхідні та основні поля (рис. 2.2).

2. Колекція userCarts. Ця колекція зберігає поточний зміст кошика користувача (рисунок 2.2). Кожен документ є кошиком окремого користувача. В колекції є ідентифікатор документа, який є ідентифікатором користувача.

3. Колекція teachers. Ця колекція розміщає в собі інформацію про всіх викладачів школи, їх опис, ім'я, фотографію та опис навичок та спеціалізацій викладача. Також кожен викладач обов'язково має свій унікальний ідентифікатор (рисунок 2.2).

4. Колекція orders. Ця колекція є досить важливою, оскільки вона розміщує інформацію про замовлення, зроблені користувачами. Кожен документ в колекції це окреме замовлення і він має свій ідентифікаційний номер. Кожен документ містить інформацію про дату та час замовлення, товар, метод оплати, загальна сума замовлення, електронна пошта та ідентифікатор користувача, який зробив замовлення.

5. Колекція courses. Це також основна колекція, оскільки вона містить всю інформацію про курси, які є на сайті школи, згруповані з мовами. Кожен документ колекції це сторінка певної мови і містить інформацію о курсах з цієї мови, а також загальну інформацію для сторінки. Кожен документ має ідентифікатор, який є кодом мови.

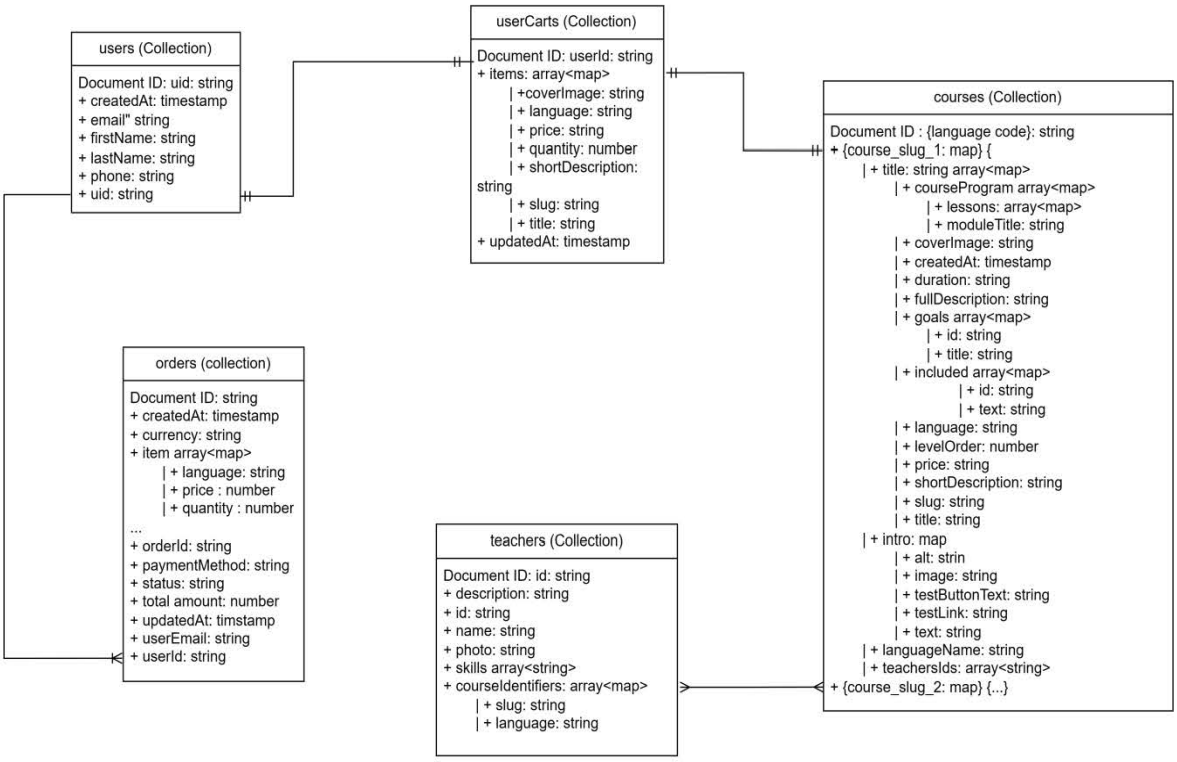


Рисунок 2.2 – Вигляд зв'язків між сутностями

Після визначення основних колекцій, документів та їх полів, наступним важливим кроком є правильне визначення зв'язків між сутностями. Вони відображають логічні взаємозалежності між об'єктами та дозволяють структуровано зберігати дані. Визначення зв'язків для нереляційної бази даних не було досить складним завданням. Результат роботи наведено на рисунку 2.2.

Роздивимося трохи детальніше зв'язки між документами в цих колекціях. За допомогою зберігання ідентифікаторів пов'язаних документів, виконується з'єднання на рівні застосунку.

– users – orders (один до багатьох), оскільки один користувач може виконувати багато замовлень;

– users – userCarts (один до одного), оскільки один користувач може мати тільки один кошик;

– orders – courses (багато до одного), оскільки кожен товар у замовленні посилається тільки на один конкретний курс;

– userCarts – courses (багато до одного), оскільки кожен товар у кошику посилається тільки на один курс;

– teachers – courses (багато до багатьох), оскільки один викладач може вести багато курсів і один курс може мати декількох викладачів.

Отже, така розроблена архітектура баз даних забезпечує ефективне та гнучке зберігання інформації про ключові компоненти системи. Структура бази даних побудовано з урахуванням специфіки NoSQL підходу. Вона є основою для подальшого розширення функціональних можливостей системи.

2.3 Візуалізація роботи системи, безпека та оцінка ризиків

Для того, щоб створити надійну та ефективну програмну систему, необхідно ретельно аналізувати всі її складові. Найважливішим етапом проєктування є створення моделей, які дозволять зрозуміти внутрішню логіку системи, передбачити можливі проблеми та оптимізувати архітектурні

рішення. Добре побудовані діаграми допомагають всім однаково побачити як буде функціонувати система. В ході проєкту було проведено аналіз поведінки та структури системи через призму різних діаграм.

1. Діаграма активностей. Діаграма активностей є однією з важливих діаграм для моделювання. Вона дозволить зрозуміти як користувач буде взаємодіяти з системою, які ключові точки прийняття рішень та які можливі варіанти розвитку подій (додаток А). На діаграмі зображено типовий сценарій поведінки користувача, де йде процес від ознайомлення зі школою до моменту придбання курсу. Це включає в себе ознайомлення зі школою, вибором мови та курсу, проходження тесту на визначення рівня та придбання курса.

2. Діаграма станів. Діаграма станів є важливою для розуміння станів системи при взаємодії користувача з інформаційною системою онлайн школи. Вона також може бути основою для реалізації маршрутів у веброзробці. Вона показує послідовність переходів між станами у процесі проходження користувачем основних кроків (рис. 2.3).

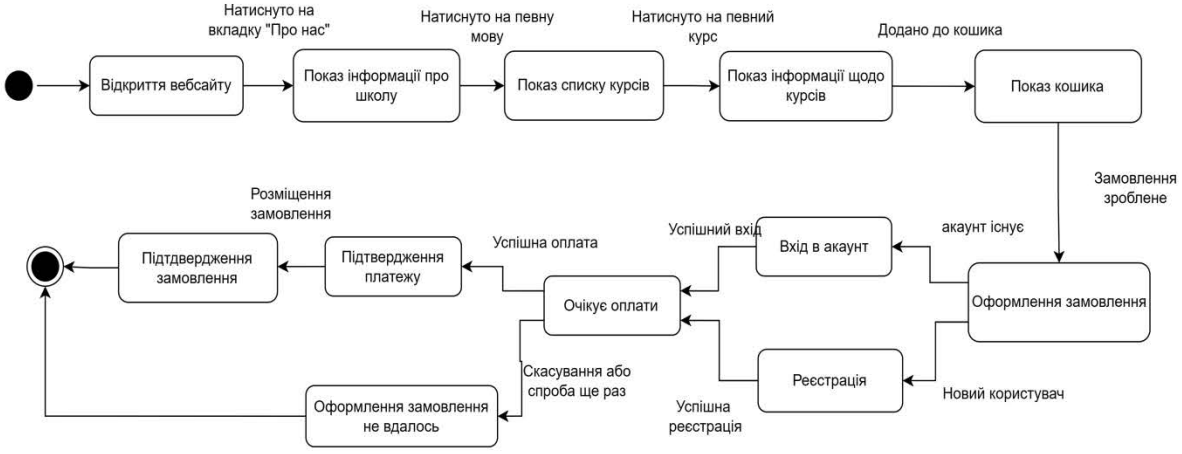


Рисунок 2.3 – Діаграма станів системи вебсайту під час основного сценарію взаємодії

Початковим станом є відкритий вебсайт. Після натискання на пункт меню система переходить до стану “показ інформації про школу”, далі “показ списку курсів”. Користувач додає курс до кошика і система

переходить у стан “показ кошика”. Для оформлення замовлення необхідно зареєструватись або увійти, і стан системи змінюється відповідно до цього. Після успішної аутентифікації система переходить до “оформлення замовлення”. У разі успішної оплати - “підтвердження платежу”, “підтвердження замовлення” та закінчення циклу. У разі помилки - “невдале оформлення”.

В цій діаграмі кожен стан системи відповідає певному кроку користувача у процесі придбання послуги в онлайн-школі. Така модель є базою для реалізації клієнтської та серверної логіки.

3. Діаграма сценаріїв взаємодії. Така діаграма є також досить важливим інструментом моделювання для описання послідовності обміну повідомленнями між об’єктами системи в рамках певного сценарію взаємодії (Додаток В).

В даному сценарії задіяно чотири основні учасники, окрім користувача. На діаграмі описано повний процес взаємодії користувача з вебсайтом від відкриття вебсайту та перегляду інформації, до сплати за замовлення. На діаграмі вказано як запити надсилаються на сервер при додаванні товару до кошика та як відбувається взаємодія з базою даних для збереження інформації про вміст кошика. Завдяки цій діаграмі можна візуалізувати поведінку системи та її складових на етапі проєктування.

4. Діаграма класів. Діаграма класів є також важливою при проєктуванні, оскільки вона відображає основні сутності предметної області та логічні зв’язки між ними. Для проєктування системи онлайн-школи було виділено класи, які охоплюють основні функціональності (Додаток Б). Таким чином, діаграма класів дуже добре допомагає зрозуміти та забезпечити гнучку архітектуру проєкту, та підтримувати її відповідно до потреб навчальної платформи.

Важливим питанням при будь-якій розробці програмного забезпечення завжди є інформаційна безпека та захист даних. Це дуже важливі аспекти для розробляємого вебсайту, оскільки система обробляє персональні дані

користувачів, платіжну інформацію та інтелектуальну власність. Отже, необхідно розглянути основні загрози інформаційній безпеці та заходи, які спрямовано на забезпечення конфіденційності та цілісності даних у системі (таблиця 2.1).

Таблиця 2.1

Основні загрози інформаційній безпеці та заходи

Питання	Які прийняти заходи
Захист персональних даних користувачів	- збір лише мінімально необхідних даних; - шифрування паролів, ніколи не зберігати їх у відкритому вигляді; - безпечна автентифікація з використанням перевірених механізмів автентифікації; - дотримання політики конфіденційності.
Безпека передачі даних	- використання протоколу HTTPS для шифрування передачі даних; - захист API за допомогою токенів автентифікації та авторизації.
Безпека бази даних	- контроль доступу до бази даних; - регулярне резервне копіювання даних; - ретельна валідація вхідних даних на серверній стороні.
Захист від поширених вебвразливостей	- захист від межсайтового скриптингу; - захист від підробки міжсайтових запитів; - безпечне управління залежностями (оновлення та перевірка сторонніх бібліотек і фреймворків).
Захист навчальних матеріалів	- авторизація доступу (матеріали лише авторизованим користувачам); - захист відео від несанкціонованого копіювання.
Моніторинг та реагування на інциденти	- журналювання подій для аналізу та розслідування; - план реагування на інциденти безпеки.

Отже, такий комплексний підхід до захисту включає технічні (шифрування, авторизація, захист API) і організаційні заходи (обмеження доступу, моніторинг інцидентів), і це дозволить забезпечити цілісність та конфіденційність інформації. А також мінімізувати ризики витоку даних, атак на сайт чи неправомірних доступів до навчальних матеріалів.

В будь-якому проекті важливим етапом розробки є також аналіз комплексу ризиків, які можуть істотно вплинути на його успішну реалізацію. Ризики можуть завадити дотриматись встановлених термінів, контролювати бюджет та забезпечити належної якості кінцевого продукту. Особливо актуальним питанням це може бути для таких систем, як онлайн-школа, яка

поєднує в собі багато різноманітних технологій та взаємодіє з багатьма користувачами. На таблиці нижче наведено аналіз основних ризиків, які можуть бути характерними для проєкту розробки системи онлайн-школи.

Таблиця 2.2

Аналіз ризиків характерних для розробляємого проєкту

Ризик	Тип	Опис
Недостатньо чіткі або змінні вимоги	Організаційний / технічний	Вимоги до функціоналу системи можуть бути не повністю визначені на початковому етапі або змінюватися в процесі розробки, що призводить до переробок та затримок.
Технічні складнощі при реалізації	Технічний	Можуть виникнути непередбачені труднощі з інтеграцією обраних технологій, реалізацією складного функціоналу або забезпеченням продуктивності.
Проблеми з безпекою даних та вразливості	Технічний / репутаційний	Недостатній рівень захисту персональних даних, платіжної інформації або навчального контенту може призвести до витоку даних, фінансових втрат та шкоди репутації.
Залежність від сторонніх сервісів	Технічний / зовнішній	Збої в роботі або зміни в API сторонніх сервісів можуть порушити функціонування системи.
Обмеженість ресурсів	Організаційний	Проект виконується в обмежені терміни однією людиною, що може призвести до неможливості реалізації запланованого функціоналу або недостатньої якості.
Проблеми з тестуванням та якістю	Технічний	Недостатнє тестування може призвести до наявності помилок та багів у фінальному продукті
Низька продуктивність системи	Технічний	При зростанні кількості користувачів або даних система може почати працювати повільно, якщо не були враховані аспекти оптимізації запитів до БД та архітектури.
Юридичні та правові аспекти	Юридичний	Недотримання законодавства про захист персональних даних, авторських прав на контент може призвести до юридичних проблем

Також необхідно роздивитись яким чином можна запобігти виникненням описаних раніше ризиків (Таблиця 2.3).

Таблиця 2.3

Ризики та можливі заходи для їх запобігання

Ризик	Можливі заходи
Недостатньо чіткі або змінні вимоги	Детальне обговорення та документування вимог на початковому етапі. Використання гнучких методологій розробки для адаптації до змін. Регулярна комунікація з замовником/керівником
Технічні складнощі при реалізації	Ретельний вибір технологій на основі аналізу їх можливостей та обмежень. Створення прототипів для складних частин системи. Залучення консультацій або додаткове вивчення документації.
Проблеми з безпекою даних та вразливості	Дотримання принципів безпечної розробки. Використання HTTPS, шифрування паролів, правил безпеки для БД. Регулярне тестування на проникнення.
Залежність від сторонніх сервісів	Вибір надійних провайдерів. Моніторинг стану сторонніх сервісів. Розробка механізмів обробки помилок при взаємодії з API. Передбачення можливих альтернатив.
Обмеженість ресурсів	Чітке планування етапів розробки. Пріоритезація функціоналу. Ефективне управління часом. Самоосвіта та пошук необхідної інформації.
Проблеми з тестуванням та якістю	Розробка тест-кейсів для основного функціоналу. Проведення модульного, інтеграційного та користувацького тестування. Використання інструментів для автоматизованого тестування.
Низька продуктивність системи	Оптимізація запитів до бази даних. Кешування даних. Вибір відповідної інфраструктури. Тестування навантаження.
Юридичні та правові аспекти	Ознайомлення з основними вимогами законодавства. Розробка політики конфіденційності та умов використання. Забезпечення можливості видалення даних користувача за запитом.

Для кожного ідентифікованого ризику також було запропоновано превентивні заходи та стратегії для того, щоб мінімізувати негативний вплив. Хоча і повне усунення всіх ризиків не можливе, їх усвідомлення та розробка відповідних заходів суттєво підвищує шанси та якісну розробку та завершення проєкту, який буде відповідати своїм цілям та вимогам користувачів.

2.4 Висновки до другого розділу

В даному розділі було проведено дослідження і обґрунтовано проєктні рішення розробки системи онлайн-школи іноземних мов. Їх результатами стали сформована концепція архітектури, обрані технологічні рішення, вибір моделі життєвого циклу і методології розробки, аналіз інформаційної

безпеки та управління ризиками. Обґрунтування вибору моделі життєвого циклу та методології розробки було на першому плані, оскільки це формує загальні принципи і стратегію розробки. Після детального аналізу переваг та недоліків поширених моделей, було прийнято рішення зупинитися на ітеративній моделі розробки, інтегрованій з принципами Agile-підходу.

Вибір технологічного стеку був здійснений з урахуванням сучасних тенденцій веброзробки, вимог до продуктивності, масштабованості та зручності подальшої підтримки системи. Для фронтенду було обрано HTML5, CSS3 та JavaScript як основні стандартні вебтехнології. Головним рішенням стало використання бібліотеки React, оскільки її архітектура компонентів дуже спростить розробку важких користувацьких інтерфейсів та підвищить ефективність коду за допомогою Virtual DOM. Такий спосіб є особливо вигідний для сайт школи, де багато елементів, які мають подібну структуру.

Для серверної логіки, зберігання даних та автентифікації було обрано платформу Firebase від Google, зокрема сервіси Firebase Firestore та Firebase Authentication. Переваги такого вибору очевидні для проєкту з малими ресурсами та часом на створення свого бекенду. Firestore забезпечує гнучкість даних, що важливо на стадії, коли потреби можуть змінюватися, і легкість розвитку в майбутньому. Firebase Authentication пропонує готові та безпечні способи для реєстрації та входу користувачів включаючи можливість зв'язку з соціальними мережами.

Для обробки онлайн-платежів було обрано інтеграцію з українською платіжною системою LiqPay, що є надійним рішенням. Середовищем розробки було обрано Visual Studio Code через його функціональність та підтримку обраних технологій. Система контролю версій Git разом із платформою GitHub забезпечить ефективно управління кодом базою.

На основі чітко визначених функціональних вимог та обраного технологічного стеку, було почато проектування архітектури та моделі даних. Для реалізації системи було створено клієнт-серверну архітектуру, де клієнтську частину було розроблено на React, а серверну логіку забезпечено

засобами Firebase. Зокрема було використано Firebase Authentification, Firestore та Hosting. Також щоб не створювати власне платіжне рішення, було інтегровано платіжну систему LiqPay, що забезпечило зручну та безпечну оплату. Структура даних у Firestore побудована у вигляді колекцій (users, userCarts, teachers, orders, courses), які логічно пов'язані між собою за допомогою унікальних ідентифікаторів. Такий підхід дозволяє легко масштабувати систему, підтримувати її ефективність і залишати простір для подальшого розвитку.

Для візуалізації основних аспектів роботи системи було використано кілька UML-діаграм. Діаграма класів допомагає зрозуміти структуру та взаємозв'язки між основними компонентами. Діаграма послідовності наочно демонструє процес взаємодії користувача з системою під час купівлі курсу, а діаграма активностей відображає бізнес-логіку вибору навчального продукту. Додатково створено загальну схему ключових етапів користувацького шляху, що дозволяє побачити проєкт із точки зору кінцевого користувача.

Також багато уваги було приділено питанням інформаційної безпеки. Було описано комплекс заходів, які спрямовано на захист персональних даних користувачів та мінімізацію різноманітних ризиків, пов'язаних з найпоширенішими вебзагрозами.

Таким чином, другий розділ заклав надійну основу для подальшої розробки вебплатформи онлайн-школи вивчення іноземних мов. Всі проаналізовані аспекти створюють чудові умови для розробки сучасного, функціонального та безпечного продукту, орієнтованого на потреби користувачів.

РОЗДІЛ 3. РОЗРОБКА САЙТУ ШКОЛИ ІНОЗЕМНИХ МОВ

3.1 Планування розробки та реалізація функціоналу

Ефективне планування є дуже важливим для успішної реалізації будь-якого програмного проєкту. Воно допомагає правильно розподілити час та ресурси, щоб зробити гарний продукт. Оскільки для розробки системи онлайн-школи було обрано ітеративну модель, то це дозволило розбити весь процес проєктування на послідовні та логічно завершені етапи. Такий підхід є дуже зручним, якщо часто змінюються вимоги та є потреба у нарощенні функціоналу.

На першому етапі необхідно було вже точно визначитись та затвердити архітектуру та модель даних для Firestore, та детальне опрацювання вимог до системи. Другий етап було присвячено реалізації основної візуальної частини сайту та наповненню статичних сторінок контентом. На третьому етапі розроблявся динамічний функціонал відображення навчальних курсів. На четвертому етапі вже було впроваджено систему автентифікації користувача, використовуючи сервіс Firebase Authentification. На п'ятому етапі було реалізовано досить важливий функціонал – кошик і формування замовлення. Найважливішим етапом комерційної складової проєкту була інтеграція платіжного модуля української платіжної системи LiqPay. І завершальним етапом була реалізація іншого залишкового або додаткового функціоналу, який не був обов'язковим. І вже після фінальної перевірки було зібрано проєкт для розгортання на Firebase Hosting.

Впровадження ключових функціональних модулів та реалізація основних розділів сайту відбувалися паралельно з вищеописаними ітераціями, де кожен модуль поступово наповнювався функціоналом та відображався у відповідних розділах сайту.

1. Модуль відображення інформаційного контенту та навігації. Він є ключовим компонентом вебплатформи, оскільки відповідає за представлення

основної інформації про школу іноземних мов, її цінності, переваги, команду та навчальні можливості. А також забезпечує зручну навігацію по всьому сайту. Головну сторінку було реалізовано більш як лендінг, який містить привітальний блок, короткий опис переваг та заклик до дії. Для структурування цих елементів було використано React-компоненти, які дозволяють гнучко керувати контентом. Ось на рисунку 3.1 приклад React-компонента для картки з описом переваг

```jsx
import React from 'react';
import './FeatureCard.css';

function FeatureCard({ imageSrc, imageAlt, description }) {
  return (
    <div className="feature-card">
      <img src={imageSrc} alt={imageAlt} className="card-image" />
      <p className="feature-description">{description}</p>
    </div>
  );
}

export default FeatureCard;
```

Рисунок 3.1 – Вигляд React-компонента для картки

Для сторінки “Викладачі” було створено динамічне відображення профілів викладачів. Було створено React-компонент “Карточка викладача” (рисунок 3.2), який відповідає за рендеринг інформації про одного викладача, і ці дані асинхронно завантажуються з колекції в Firestore.

```jsx
function TeacherCard({name, photo, description, skills}) {
    return (
        <div className="teacher-card">
            <div className="teacher-image-container">
                <img src={photo} alt={name} />
                <h3>{name}</h3>
            </div>
            <div className="teacher-info">
                <p className='teacher-description'>{description}</p>
                <ul className='teacher-skills'>
                    {skills.map((skill, index) => (
                        <li key={index}>{skill}</li>
                    ))}
                </ul>
            </div>
        </div>
    )
}
```

Рисунок 3.2 – React-компонент для рендеру інформації однієї картки

Сторінка курсів це головний розділ, оскільки він представляє навчальні програми школи. Цю сторінку було створено динамічно для завантаження інформації про доступні мови з колекції courses. При виборі мови відбувається завантаження та відображення списку курсів для цієї мови. А для забезпечення плавної та інтуїтивної навігації між розділами сайту було використано бібліотеку React Router та налаштовано маршрути для всіх основних сторінок.

2. Модуль управління кошиком та замовленнями. Він є критично важливим для комерційної складової цього проєкту. Він забезпечить можливість вибору курсів, формування замовлень та їх оплати.

До розробленого функціоналу кошика входять додавання обраного курсу, видалення його з кошика або зміну кількості. Інформація про вміст кошика зберігається в колекції userCarts у Firestore, де документом є userId. В кошику відображається список доданих послуг, ціна та загальна сума (рис. 3.3).

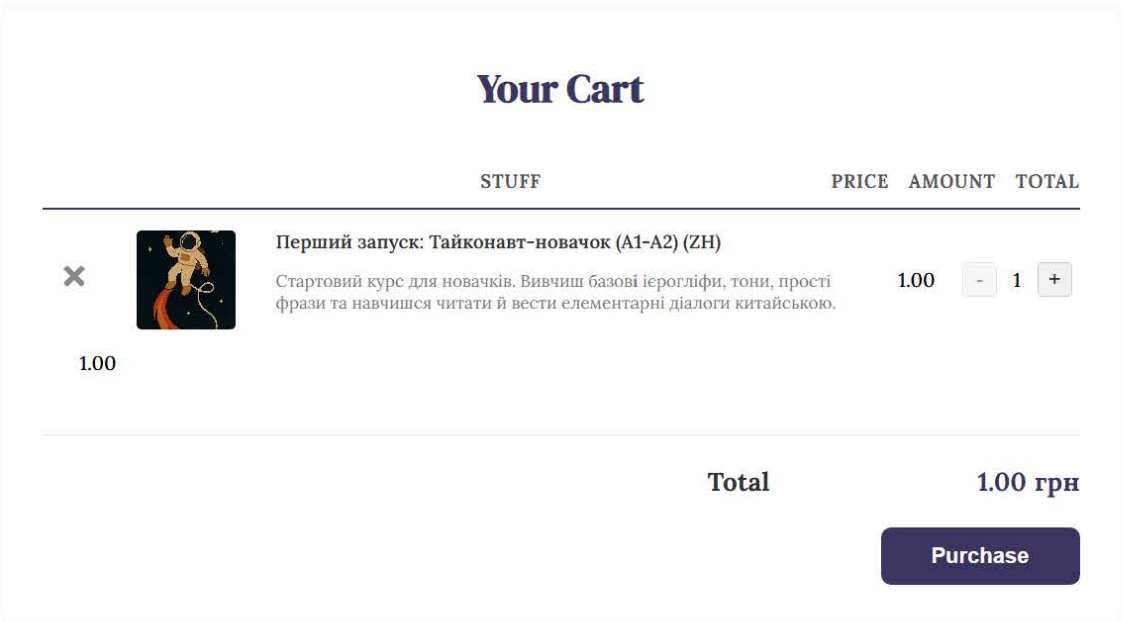


Рисунок 3.3 – Вигляд кошика з обраною послугою

Після переходу з кошика до оформлення замовлення, система ініціює створення нового документа в колекції orders. Якщо користувач не

автентифікований, йому пропонується увійти або зареєструватися. Для автентифікованого користувача відбувається збір необхідної інформації та додаткової інформації, та підтвердження замовлення.

3. Інтеграція платіжного модуля. Це було найскладнішою частиною, забезпечити можливість онлайн-оплати за допомогою інтеграції з LiqPay. Після підтвердження замовлення на стороні сервера, генерується запит до API LiqPay з параметрами платежу. Після цього клієнта перенаправляє на захищену сторінку LiqPay, де він здійснює оплату. Після успішної оплати або помилки оплати, система перенаправляє користувача на відповідну сторінку (рис. 3.4).

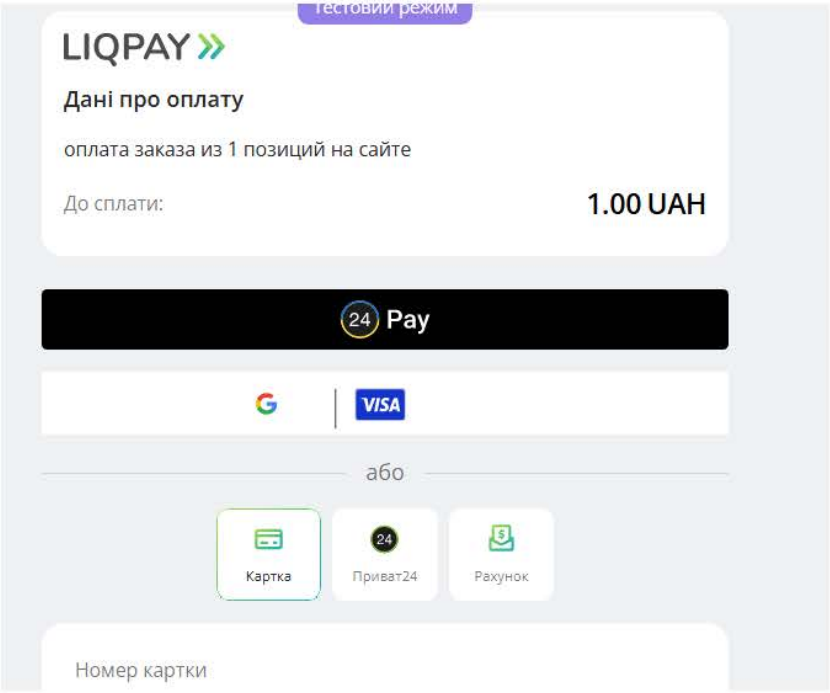


Рисунок 3.4 – Перенаправлення користувача на сторінку оплати

4. Інтерактивні елементи та користувацький досвід. Окрім основного функціоналу, увага приділялася створенню позитивного користувацького досвіду. Весь сайт розроблено з використанням принципів адаптивного дизайну. Елементи інтерфейсу адаптуються до розміру екрану, забезпечуючи читабельність та легкість навігації (рис. 3.5).

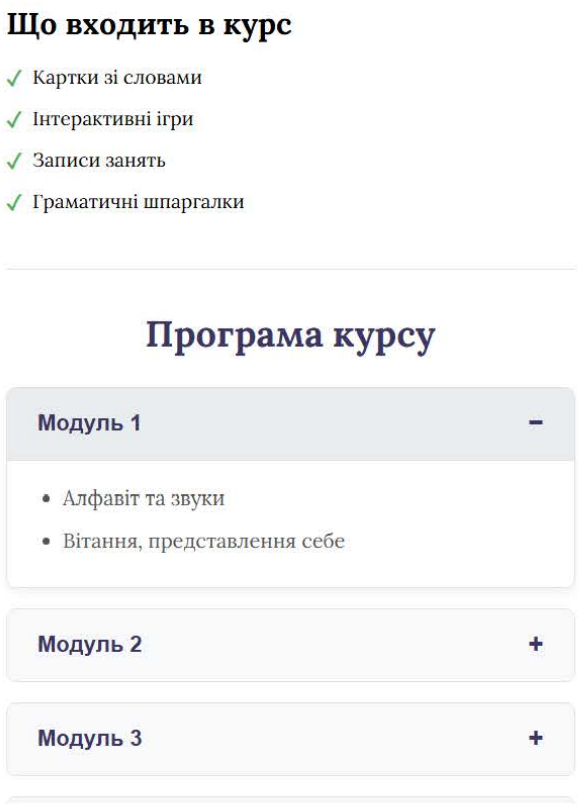


Рисунок 3.5 – Вигляд сайту на пристрою з меншим розширенням екрану

У системі школи було реалізовано стандартну процедуру реєстрації користувачів за допомогою електронної пошти та пароля. Для цього було створено окрему форму реєстрації, де також треба ввести ім'я і прізвище. Після заповнення форми, дані надсилаються до бази даних і у колекції users створюється окремий запис із зберіганням профільних даних, прив'язаних до унікального ідентифікатора користувача, наданого Firebase. На рис. 3.6 наведено запис з бази даних про зареєстрованого користувача.

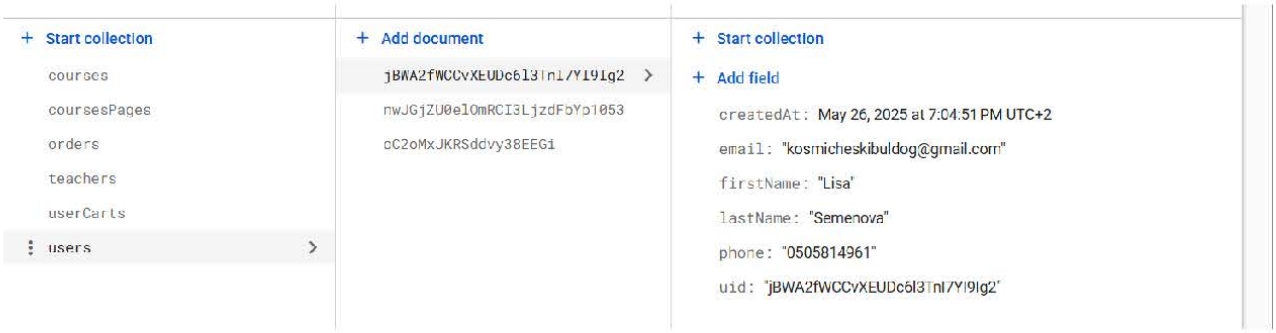


Рисунок 3.6 – Вигляд документа про користувача в базі даних

Для входу в систему було реалізовано окрему форму автентифікації, де необхідно ввести електронну пошту та пароль. Окрім базової автентифікації, передбачено також функціонал виходу з системи, який викликає відповідний метод Firebase Authentication і припиняє дію сесії користувача. На рисунку 3.7 наведено форму входу до системи

Рисунок 3.7 – Вигляд форми входу до системи

Таким чином, за допомогою Firebase Authentification можна впровадити надійну, масштабовану та легку в обслуговуванні системи автентифікації. Вона буде відповідати актуальним вимогам безпеки та забезпечувати зручну взаємодію користувача з системою.

3.2 Візуальне оформлення, тестування та надійність системи

Успішність програмного продукту визначається не лише обсягом реалізованих функцій, але й тим, наскільки зручно, інтуїтивно та приємно користувачу взаємодіяти з інтерфейсом, наскільки стабільно система працює в реальних умовах, і як вона поводиться у випадку нестандартних або критичних ситуацій. Саме тому під час розробки онлайн-платформи для

школи іноземних мов значну увагу було приділено як зовнішньому вигляду інтерфейсу, так і технічній стабільності системи.

Інтерфейс вебсайту школи іноземних мов було розроблено з акцентом на створенні сучасного, інтуїтивно зрозумілого та естетично привабливого користувацького досвіду. До основних аспектів візуальної реалізації включають:

1. Дизайн та брендинг. Для сайту було розроблено фірмовий космічний стиль, що включає специфічну колірну палітру, типографіку та логотип, який відповідає тематиці освітньої платформи. Колірна схема підібрана таким чином, щоб бути приємною для сприйняття та не відволікати від основного контенту, також вона відповідає основній тематиці проєкту.

Стосовно типографіки, то було обрано чіткі та читабельні шрифти для основного тексту і навігаційних елементів, що створює баланс між академічністю та доброзичливістю.

Логотип сайт підкреслює характер школи, він являє собою відкритість до нових знань, сучасність та дружність до користувача (рис. 3.8).

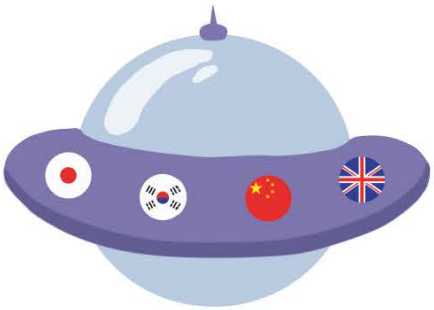


Рисунок 3.8 – Вигляд логотипу вебсайту

2. Структура та макет сторінок. Структура всіх сторінок сайту є послідовною та логічною, що полегшує орієнтацію користувачів. Основні навігаційні елементи, як хедер та футер залишаються консистентними на всіх сторінках. І всі макети сторінок було розроблено з урахуванням ієрархії інформації, де всі найважливіші елементи розміщені на видних місцях (рис. 3.9).

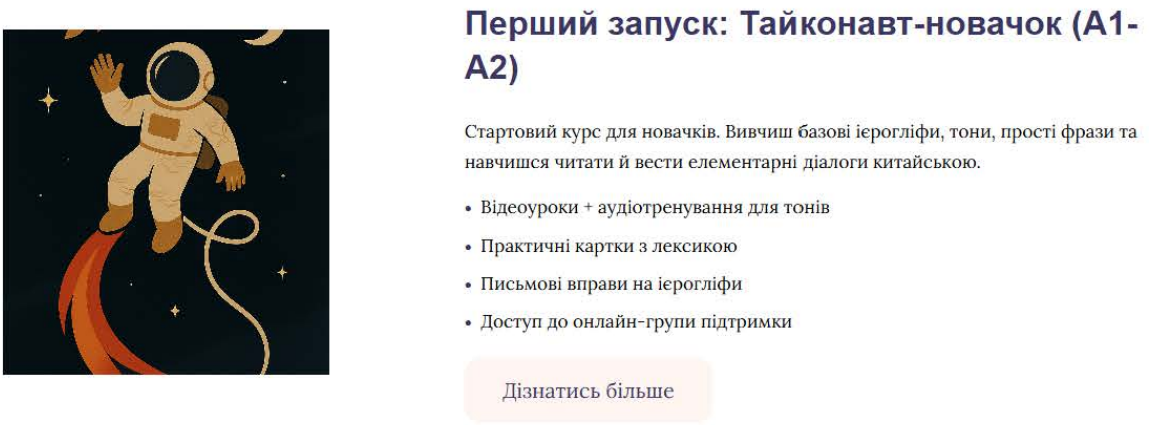


Рисунок 3.9 – Кнопка заклику до дії

3. Інтерактивні елементи та анімації. Використання CSS анімацій та переходів покращує візуальне сприйняття та зворотного зв'язку на дії користувача. Інтерактивні елементи, такі як слайдери, випадаючі меню, модальні вікна, реалізовані таким чином, щоб бути інтуїтивно зрозумілими та не перевантажувати інтерфейс. На рис. 3.10 зображено анімацію кнопки, при наведенні вона змінює колір.

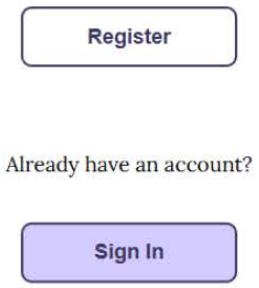


Рисунок 3.10 – Вигляд анімації кнопки входу до акаунту

4. Візуалізація ключових сторінок. Далі необхідно перевірити працездатність функціоналу, оскільки тестування є невід'ємною частиною процесу розробки, спрямованою на виявлення та виправлення помилок. А також на перевірку відповідності реалізованого функціоналу заявленим вимогам. У рамках даного проєкту було застосовано наступні види тестування:

1. Модульне тестування, де тестувались окремі компоненти React та функції JavaScript на коректність їх роботи ізольовано від інших частин системи. Перевірялась правильність рендерингу компонентів при різних вхідних даних.

2. Інтеграційне тестування перевіряло взаємодію між різними компонентами та модулями системи. Наприклад, було проведено тестування процесу додавання товару до кошика та оновлення його стану (рис. 3.11).

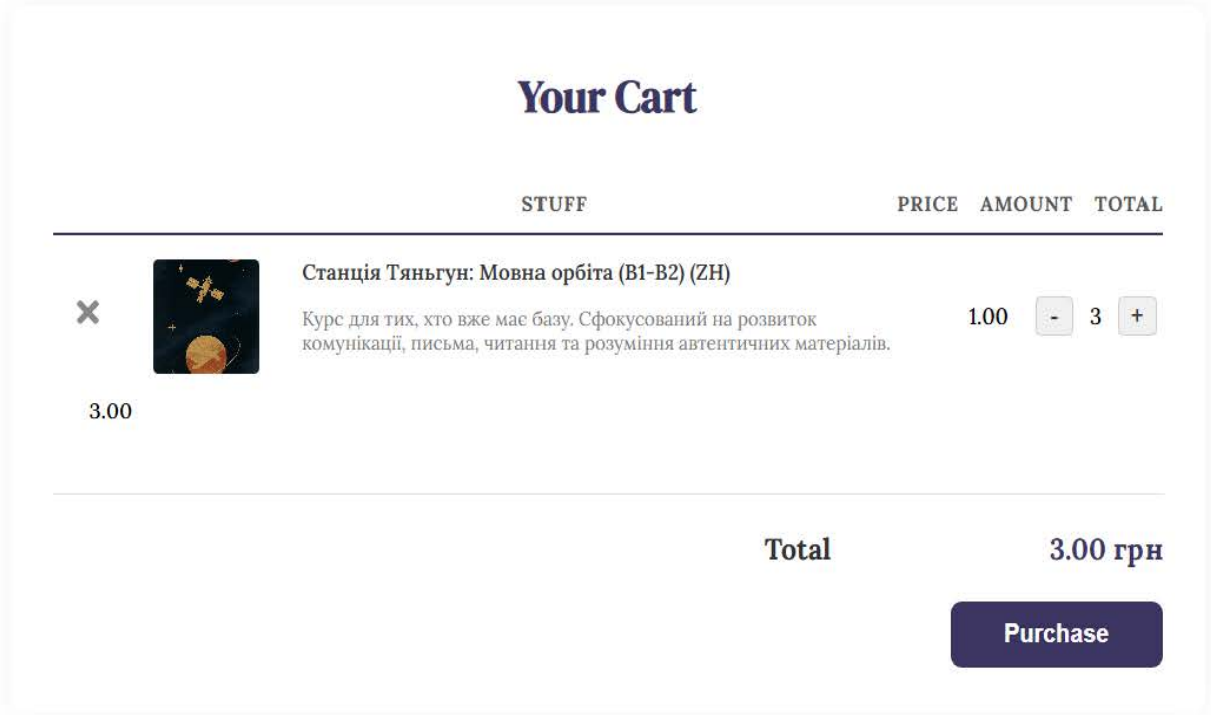


Рисунок 3.11 – Вигляд кошику та доданих товарів

3. Функціональне тестування на рівні користувацьких сценаріїв перевіряє працездатність ключових користувацьких сценаріїв від початку до кінця. Зазвичай це ручне тестування, яке імітує дії реального користувача.

До основних протестованих сценаріїв належать: реєстрація та автентифікація користувача, робота з кошиком, оформлення замовлення, коректність роботи навігації по сайту. На рис. 3.12 наведено приклад тестування автентифікації користувача при вводу неправильної інформації.

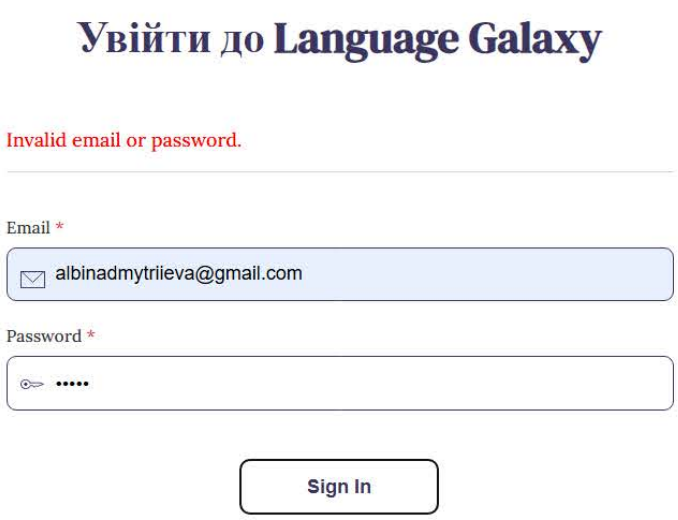


Рисунок 3.12 –Ввід неправильної інформації користувача

4. Тестування користувацького інтерфейсу перевіряє коректність відображення всіх елементів інтерфейсу, їх відповідність дизайну, зручності використання та адаптивності на різних екранах та браузерах. В ході тестування було виявлено та виправлено низку помилок верстки та логіки окремих функцій, як, наприклад навіть при успішному завершенню оплати, при поверненні на вебсайт, система показувала помилку (рис. 3.13) .

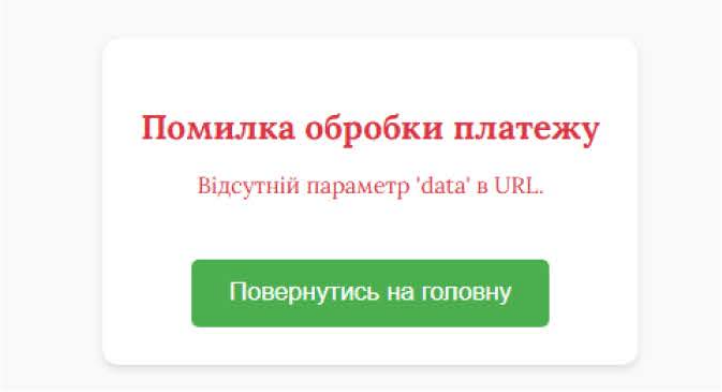


Рисунок 3.13 – Помилка логіки функції

Отже, основний функціонал системи працює коректно, та відповідає усім заявленим вимогам та очікуванням користувачів. Більш детальний звіт про тестування наведено в додатку.

Що стосується надійності та відмовостійкості системи, то вони визначають її здатність коректно функціонувати протягом тривалого часу та адекватно реагувати на можливі помилки. Особливо важливим це є у вебдодатків, орієнтованих на кінцевого користувача, оскільки можуть обробляти непередбачувані події без суттєвих збоїв та швидко відновлювати працездатність у разі помилок. У таблиці наведено основні аспекти та реалізовані заходи для підвищення надійності та відмовостійкості системи.

Таблиця 3.1

Аспекти надійності та реалізовані для них заходи

Аспект надійності / відмовостійкості	Реалізовані заходи	Очікуваний результат
1. Валідація вхідних даних	Клієнтська валідація форм (обов'язкові поля, формат email, довжина пароля тощо) з використанням JavaScript/React. Серверна валідація через правила безпеки Firestore та логіку Firebase Functions.	Запобігання запису некоректних даних, зменшення навантаження на серверну частину, покращення UX, забезпечення цілісності даних у БД.
2. Обробка помилок	Обробка помилок на клієнті з відображенням інформативних повідомлень. Обробка різних статусів відповіді від платіжної системи LiqPay.	Стабільна робота додатку при виникненні помилок, надання користувачеві зрозумілого зворотного зв'язку, запобігання "падінню" системи.
3. Fallback-механізми	Спеціальна сторінка "404 - Не знайдено".	Покращення користувацького досвіду при навігації на неіснуючі сторінки або при критичних збоях.
4. Реакція на недоступність сервісів	Обробка помилок з'єднання з Firebase, інформування користувача. Обробка помилок при взаємодії з LiqPay, інформування користувача.	Мінімізація негативного впливу на користувача при тимчасовій недоступності ключових зовнішніх сервісів.

1. Валідація вхідних даних. Дуже важливим моментом розробки програмного забезпечення є забезпечення коректності та безпеки даних, які користувачі вводять в систему. Якщо валідація відсутня або погана, то це

може призвести до серйозних проблем. В цьому проєкту було реалізовано багаторівневий підхід до валідації даних.

Клієнтська валідація перевіряє чи заповнені всі обов'язкові поля (рисунок 3.14), контролює формат даних та дотримання специфічних обмежень.

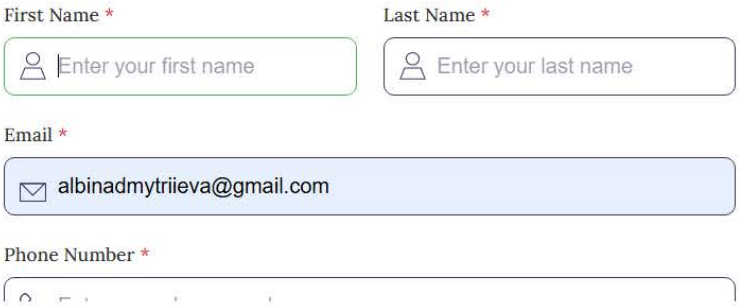


Рисунок 3.14 – Перевірка на заповненість полів

Також, наприклад, система автоматично перевіряє чи містить поле електронної пошти необхідні символи та чи має правильну структуру, як на рисунку 3.15.

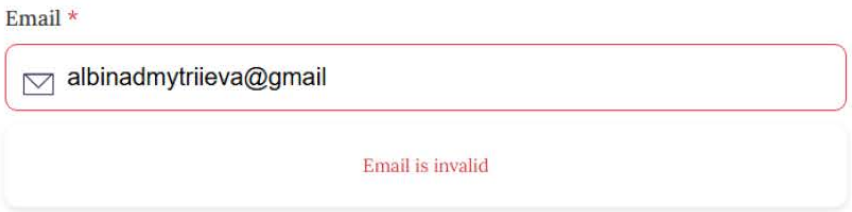


Рисунок 3.15 – Перевірка правильності вводу пошти

Другий рівень валідації це серверна валідація, яку реалізовано через систему правил безпеки Firebase Firestore. Вони визначають права доступу до даних, їх структуру та допустимі типи. На рисунку 3.16 наведено правила які було написано для полегшення тестування на етапі розробки. Видно, що для колекції userCarts доступ на читання та запис до документа кошика надається лише тому користувачеві, який є його власником.

```
rules_version = '2';

service cloud.firestore {
  match /databases/{database}/documents {

    // This rule allows anyone with your Firestore database reference to view, edit,
    // and delete all data in your Firestore database. It is useful for getting
    // started, but it is configured to expire after 30 days because it
    // leaves your app open to attackers. At that time, all client
    // requests to your Firestore database will be denied.
    //
    // Make sure to write security rules for your app before that time, or else
    // all client requests to your Firestore database will be denied until you Update
    // your rules
    match /{document=**} {
      allow read, write: if request.time < timestamp.date(2025, 6, 20);
    }

    match /userCarts/{userId} {
      allow read, write: if request.auth != null && request.auth.uid == userId;
    }
  }
}
```

Рисунок 3.16 – Набір правил з Firebase Firestor

2. Обробка помилок та виняткових ситуацій. Система повинна передбачати можливість виникнення різноманітних помилок та все одно забезпечити їх коректну обробку без впливу на додаток. Для перехоплення та обробки помилок на стороні клієнта застосовуються стандартні механізми JavaScript (рисунок 3.17). У разі виникнення будь-якої помилки, користувач отримує чітке повідомлення.

```
try {
    console.log(`[PaymentResultPage] Fetching status for orderId: ${processingOrderId}`);
    const statusResult = await getPaymentStatus(processingOrderId);
    setPaymentStatusInfo(statusResult);
    console.log("[PaymentResultPage] Status from backend:", statusResult);

    if (statusResult && (statusResult.status === 'success' || statusResult.status === 'sand
    } else if (statusResult) {
        console.warn(`[PaymentResultPage] Платіж для замовлення ${processingOrderId} не усп
        if (statusResult.status === 'failure' || statusResult.status === 'error' || statusR
        }
    } else {···
    }
} catch (err) {
    console.error('[PaymentResultPage] Помилка при запиті статусу платежу з бэкенду:', err)
    setError(err.message || 'Сталася помилка під час перевірки статусу платежу.');
```

Рисунок 3.17 – Приклад блока try…catch в коді

3. Fallback-механізм. При переході на неіснуючі посилання або при виникненні непередбачених глобальних помилок, було розроблено сторінку з повідомленням про помилку та з пропонуванням навігаційних посилань для повернення на головну сторінку чи інші, як на рис. 3.18.

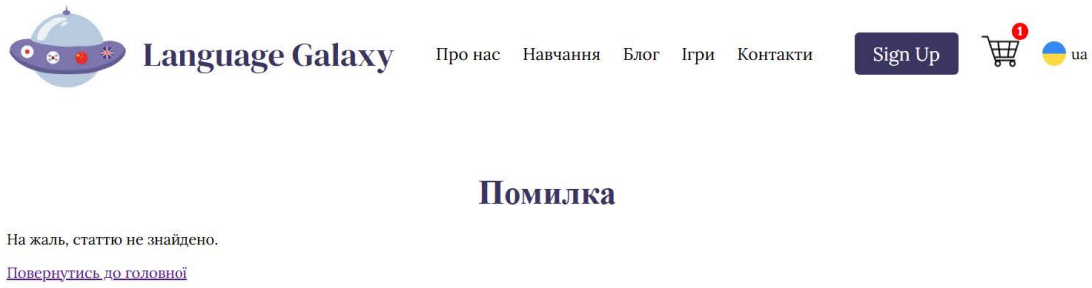


Рисунок 3.18 – Вигляд кастомізованої сторінки помилки

4. Реакція на недоступність зовнішніх сервісів. Оскільки вебсайт тісно пов’язаний з Firebase, то в разі його тимчасової недоступності, вебсайт буде суттєво обмежено в функціональності. Але користувачеві буде відображено повідомлення, якщо неможливо виконати операцію або завантажити дані. А якщо платіжна система LiqPay буде тимчасово недоступна, то користувач не зможе здійснити оплату і система повідомить його про неможливість проведення транзакції.

3.3 Підготовка до експлуатації, підтримка та оцінка якості

Отже, останніми етапами розробки програмного продукту є підготовка до реальної експлуатації, планування заходів з технічної підтримки та об’єктивна оцінка якості створеної системи. Це все є дуже важливим для того, щоб вебсайт онлайн-школи іноземних мов ефективно працював та задовольняв потреби користувачів.

Тож, перед запуском системи в експлуатацію та щоб забезпечити стабільну роботу в майбутньому було враховано наступні аспекти.

1. Хостинг та розгортання. Як ж зазначалось раніше, то для розгортання клієнтської частини React-застосунку та для обслуговування статичних файлів було вирішено використовувати сервіс Firebase Hosting. Його було вибрано оскільки він має тісну інтеграцію з іншими корисними сервісами Firebse, автоматичним наданням SSL-сертифікату для забезпечення HTTPS-з'єднання та він досить простий в налаштуванні та розгортанні.

2. Оновлення контенту. Більшість контенту, як інформація про курси, викладачів та користувачів зберігається в базі даних Firestore. Це дозволяє досить зручно оновлювати контент безпосередньо через консоль Firebase. Але весь інший контент, який не зберігається в базах даних, буде вимагати оновлення відповідних React-компонентів та втручання в код.

3. Резервне копіювання даних. Для того, щоб мати можливість відновити дані у випадку непередбачених збоїв або пошкодження інформації, було налаштовано регулярні автоматизовані процедури експорту даних Firestore до служби Google Cloud Storage. Що стосовно коду, то кодова база проєкту зберігається в системі Git, що вже є механізмом резервного копіювання коду.

4. Моніторинг та технічна підтримка. Дуже зручно, що Firebase вже має інструменти для моніторингу використання сервісів та виявлення помилок, як, наприклад Firebase Functions для логування. Щодо технічної підтримки, то користувачі можуть звертатись через електронну пошту для вирішення певних питань. Повідомлення будуть аналізуватись для вирішення та вдосконалення проблем в майбутньому.

Також, наступним дуже важливим етапом є визначення оцінки якості продукту, тобто зрозуміти наскільки він відповідає початковим вимогам, очікуванням користувачів та загальноприйнятим стандартам (Додаток Г).

Отже, на основі проведеного аналізу можна зробити висновок, що розроблений програмний продукт відповідає поставленим функціональним вимогам, є надійним, зручним у використанні та демонструє належну

ефективність. Незначні недоліки, які було виявлено в ході тестування, було усунено і це дозволяє позитивно оцінити загальну якість реалізованого програмного продукту.

Також для того, щоб забезпечити ефективне використання розробленого вебсайту онлайн-школи іноземних мов, було підготовлено коротку інструкцію для користувача, яка охоплює основні сценарії взаємодії з системою.

Для того, щоб зареєструватись на сайті, користувачеві необхідно перейти на головну сторінку сайту, натиснути кнопку “Реєстрація” в меню. Відкриється реєстраційна форма, де користувачу треба буде заповнити поля ім’я і прізвище, електронну адресу, номер телефону, пароль та підтвердити пароль. Натиснути “зареєструватись” і користувач вже с системі.

Для того, щоб увійти в систему, користувачеві треба натиснути кнопку “Вхід” у верхньому меню, ввести електронну адресу і пароль, та натиснути кнопку “Увійти”.

Для того, щоб обрати курс, необхідно перейти до розділу “Навчання” через головне меню та обрати мову, яка цікавить користувача. Або ж в випадаючому меню одразу обрати мову. На сторінці буде відображено список доступних курсів, користувачеві треба бути натиснути на картку курсу для перегляду більш детальної інформації. Також якщо користувач не впевнений у тому, який рівень курсу обрати, він може пройти тестування на сайті для визначення рівня володіння мовою.

На сторінці детального опису курсу користувачеві необхідно натиснути кнопку “Додати до кошика” і з’явиться повідомлення про успішне додавання та також оновиться іконка кошика.

Для того, щоб оформити замовлення, користувачеві треба перейти до кошика, натиснувши іконку в головному меню та натиснути кнопку “Придбати”. Якщо користувач ще не увійшов в систему, йому буде запропоновано увійти або зареєструватись. Після цього його буде перенаправлено на захищену сторінку платіжної системи LiqPay, де треба

буде ввести платіжні дані та підтвердити оплату. Після завершення оплати користувача буде повернено на сайт школі зі сповіщенням про результат транзакції.

Після успішної оплати з користувачем зв'яжеться менеджер та надасть доступ до обраних курсів.

3.4 Висновки до третього розділу

Третій розділ кваліфікаційної роботи було присвячено детальному опису вже практичної реалізації програмного продукту, а саме вебсайту для онлайн школи іноземних мов. В цьому розділі було описано основні етапи планування розробки, впровадження ключових функціональних модулів, створення візуального інтерфейсу, проведення тестування, забезпечення надійності системи, підготову до експлуатації та фінальну оцінку якості.

Основою процесу розробки були планування та ітеративна реалізація функціоналу. Туди входив вибір ітеративної моделі, доповненої принципами Agile. Цей пункт дозволив ефективно управляти проєктом, розбиваючи його на ітерації та забезпечувати можливість адаптації до змін.

Важливим кроком також була реалізація модуля відображення інформаційного контенту та навігації. Треба було створити основні статичні сторінки з використанням HTML5, CSS3 та розробити базові React-компоненти, які б зустрічались протягом всього сайту. Для того, щоб забезпечити плавні переходи між сторінками було інтегровано бібліотеку React Router. Далі для динамічної інформації було реалізовано спеціалізовані React-компоненти, дані для яких асинхронно завантажуються з відповідних колекцій у Firestore.

Необхідно було впровадити модуль управління кошиком та замовленням. Зайняло трохи часу для розробки функціоналу, що дозволило би користувачам додавати обрані курси до кошика, переглядати його вміст,

змінювати кількість товарів. Важливо було зробити так, щоб стан кошика авторизованих користувачів синхронізувався та зберігався в базі даних.

Найскладнішим та найважливішим етапом була інтеграція платіжного модуля з українською платіжною системою LiqPay. Було важливо правильно реалізувати механізм формування запиту до API LiqPay та безпечне перенаправлення користувача на захищену платіжну сторінку. Завдяки Firestore при успішній або неуспішній оплаті оновлення статусу фіксується в базі даних, а користувач отримує відповідне сповіщення.

За допомогою сервісу Firebase Authentification було реалізовано безпечний та зручний доступ до системи через засоби ідентифікації та автентифікації. Було розроблено форми та логіки входу та реєстрації нових користувачів на вебсайті. Також було додано функціонал виходу із системи.

Також одним із важливих аспектів було створення позитивного користувацького досвіду та інтерактивності на сайті. Для цього використовувались принципи адаптивного дизайну, інтерактивні елементи як слайдери та випадаючі меню, плавні анімації та переходи покращили візуальне сприйняття інтерфейсу.

Для сайту було вирішено розробити унікальний та фірмовий стиль “космічної тематики”, який включає відповідну специфічну колірну палітру, відповідну назву та логотип. Все це відповідає освітній спрямованості платформи та створює приємне візуальне враження.

Також одним із найважливіших етапів було проведено комплексне тестування, щоб перевірити на скільки працездатна та якісно реалізована система. Це тестування було важливим, оскільки дало змогу перевірити коректність роботи окремих React-компонентів та JavaScript функцій. Важливо було перевірити на скільки добре взаємодіють клієнтська частина з сервісами Firebase. Було використано також тестування на рівні користувацьких сценаріїв, що підтвердило працездатність всіх основних процесів. В ході тестування було виявлено декілька помилок та недоліків, але вдалось їх виправити та забезпечити гарний вигляд в браузерах.

Для того, щоб забезпечити надійність та відмовостійкість системи було вирішено впровадити багаторівневу валідацію вхідних даних. Щоб система нормально реагувала на нестандартні ситуації, було реалізовано механізм обробки помилок на клієнтській та серверній сторонах. Було передбачено fallback-механізми та продумано поведінку системи у випадку якоїсь тимчасової недоступності інтегрованих сервісів.

І останнім етапом було планування підготовки до експлуатації та технічної підтримки. На даному етапі необхідно було розгорнути сайт на Firebase Hosting, продумати стратегії оновлення контенту та налаштувати резервне копіювання даних. Обов'язково було проведено оцінку якості програмного продукту за критеріями функціональності, надійності, зручності користування, ефективності та безпеки. Це все підтвердило те, що розроблена система відповідає всім поставлених вимогам та готова до використання.

Отже, у третьому розділі показано, як проєктні рішення, викладені в другому розділі, були втілені у готовий, функціональний та випробуваний програмний продукт. Поетапна та ітеративна розробка, прискіпливе тестування кожного окремого модуля, а також чітко сплановані заходи з гарантування надійності та зручності використання сприяли створенню вебсайту онлайн-школи іноземних мов. Цей сайт технічно завершений, відповідає сучасним вимогам веброзробки та повністю підготовлений до подальшого розвитку та безпосереднього впровадження в практичне використання.

ВИСНОВКИ

Дана кваліфікаційна робота присвячена комплексному вирішенню актуальної задачі проєктування та розробки програмного застосунку – вебсайту для онлайн-школи іноземних мов. Метою роботи було створення сучасного, функціонального, безпечного та зручного для користувача вебресурсу, який би забезпечував ефективну взаємодію між студентами, викладачами та адміністрацією школи, а також надавав можливості для вибору та оплати навчальних курсів. Для досягнення поставленої мети було вирішено певну низку завдань, що включало в себе завдання від аналізу предметної області та формування вимог, до вибору технологічного стеку, проєктування архітектури, безпосередньої реалізації функціоналу, тестування та підготовки до експлуатації.

На першому етапі роботи було проведено глибокий аналіз предметної області онлайн-освіти, вивчено функціональні можливості та технологічні рішення існуючих платформ для вивчення іноземних мов. Це дозволило чітко сформулювати функціональні та нефункціональні вимоги до розроблюваної системи. Актуальність проєкту підтверджується стійким зростанням ринку онлайн-навчання та потребою в якісних, доступних та технологічно розвинених освітніх платформах.

На другому, проєктному, етапі було здійснено обґрунтування вибору ключових архітектурних та технологічних рішень. Було вирішено обрати ітеративну модель життєвого циклу, яка інтегрована з принципами Agile-методології. Така модель дозволяє забезпечити гнучкість для адаптацій до можливих змін у вимогах та дозволить впроваджувати функціонал поетапно. Оскільки ресурси були обмежені та стислі терміни кваліфікаційного проєктування, така модель виявилась досить ефективною.

При виборі технологічного стеку було враховано на скільки сучасний, продуктивний та масштабований додаток необхідно створити. Для клієнтської частини було вирішено обрати бібліотеку React, оскільки вона

має зручну компонентну архітектуру та Virtual DOM. Використання сучасних технологій HTML5, CSS3 та JavaScript забезпечило стандартний набір інструментів. Також для того, щоб зробити адаптивний дизайн було використано "чистий" CSS, без готових фреймворків, оскільки таким чином це надає повний контроль над візуальним стилем. Також було інтегровано бібліотеку React Router для управління навігацією.

Також одним з основних прийнятих рішень було рішення використовувати платформу Firebse для серверної інфраструктури та управління даним. Перевагою такого підходу (Backend-as-a-Service) є значне скорочення часу на розробку та підтримку власної серверної інфраструктури, висока надійність, масштабованість та вбудовані механізми безпеки, що надаються Google. А інтеграція з українською платіжною системою LiqPay забезпечила можливість реалізації онлайн-оплати курсів, що є важливою комерційною складовою проєкту.

Далі для візуалізації архітектури та поведінки системи було розроблено набір UML діаграм. Діаграма класів детально описала статичну структуру програмних компонентів, їх атрибути, методи та взаємозв'язки. Діаграма послідовності проілюструвала динаміку взаємодії об'єктів під час виконання ключового сценарію покупки курсу. Діаграма активностей візуалізувала логіку основного бізнес-процесу з точки зору користувача. Схема ключових етапів взаємодії надала високорівневий огляд користувацького шляху. Ці моделі сприяли кращому розумінню системи та узгодженню проєктних рішень.

Також багато уваги було приділено питанням інформаційної безпеки. Було проаналізовано потенційні загрози та визначено заходи для захисту персональних даних користувачів, забезпечення безпеки передачі даних (використання HTTPS), захисту бази даних та протидії поширеним вебвразливостям. Було проведено також аналіз ризиків реалізації проєкту, що охопив організаційні, технічні, зовнішні та юридичні аспекти, та запропоновано стратегії їх мінімізації.

На третьому, практичному, етапі було здійснено безпосередню розробку та реалізацію програмного продукту. Поетапно, відповідно до обраної ітеративної моделі, було впроваджено ключові функціональні модулі: модуль відображення інформаційного контенту та навігації (головна сторінка, сторінки "Про школу", "Викладачі", "Курси"); модуль управління кошиком та замовленнями; інтегровано платіжний модуль LiqPay; реалізовано систему ідентифікації та автентифікації користувачів за допомогою Firebase Authentication. Також було розроблено адаптивний дизайн та інтерактивні елементи для того, щоб покращити користувацький досвід.

Далі було проведено комплексне тестування системи, яке включало в себе модульне, інтеграційне, функціональне тестування, тестування користувацького інтерфейсу, адаптивності та кросбраузерності. Ці тестування дозволили виявити деякі існуючи недоліки та усунути їх, щоб забезпечити правильну роботу основного функціоналу. Після оцінки якості програмного продукту за багатьма важливими критеріями, можна було наголосити на тому, що система відповідає поставленим вимогам. Особливо що React та Firebase Hosting забезпечили швидкодію та хорошу супроводжуваність коду. Також для користувача було розроблено інструкцію для полегшення освоєння системи.

Розроблений в рамках кваліфікаційної роботи вебсайт онлайн-школи іноземних мов це комплексне технологічне рішення, яке успішно реалізувало поставлені цілі та завдання на початку роботи.

Створений продукт має багато переваг. До першої з них треба віднести добре продуманий вибір технологічного стеку та технологій, як використання React та Firebase. Це забезпечило високу продуктивність, масштабованість та можливість подальшого розвитку вебсайту. Завдяки компонентному підходу React та використанню нереляційної бази даних Firestore, можна легко модифікувати та розширювати функціонал системи.

Не дивлячись на всі переваги, розроблений вебсайт також має деякі недоліки. З основних недоліків це залежність від Firebase. Хоча і сервер

надає багато переваг, повна залежність від нього може становити деякий ризик у довгостроковій перспективі, як, наприклад зміна тарифів або умов використання. Також, на даному етапі, для сайту відсутня повноцінна адміністративна панель, тому управління контентом через консоль Firebase може бути не досить зручним для нетехнічного персоналу школи. Також може виникнути деяка складність реалізації деяких специфічних функцій без власного бекенду.

Отже, як було з'ясовано, то платформа має шляхи подальшого розвитку та модернізації. Говорячи про майбутнє, першою чергою можна бути розробити повноцінну адміністративну панель, щоб було зручно управляти всіма аспектами платформи. Також досить важливим етапом буде реалізація особистого кабінету студента та викладача, де буде доступ до навчальних матеріалів, можна буде відстежувати прогрес та спілкуватись онлайн з іншими студентами. Пізніше можна буде розробити мобільний додаток, щоб забезпечити ще більшу доступність та зручність навчання.

Новизна даної роботи полягає у комплексному підході до проєктування та реалізації сучасної освітньої вебплатформи. Особливістю запропонованого рішення є інноваційне поєднання гнучкості компонентної архітектури React з потужністю хмарних сервісів Firebase, що створює оптимальний баланс між функціональною повнотою системи та простотою її розробки і підтримки. Особливо цінним є досвід успішної інтеграції з українською платіжною системою LiqPay, що робить створене рішення максимально адаптованим до специфіки вітчизняного ринку освітніх послуг.

Отже, підсумовуючи результати проведеного дослідження, можна сказати, що всі поставлені цілі та завдання на початку роботи були успішно досягнуті. Було створено функціональний, зручний та ефективний вебсайт онлайн школи вивчення іноземних мов, який відповідає сучасним вимогам користувачів та забезпечує потреби школи у презентації своїх послуг, залученні нових учнів та автоматизації процесів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Global Online Education Market Report 2022, Grand View Research. URL: https://www.grandviewresearch.com/industry-analysis/education-technology-market

2. Frain B. Responsive Web Design with HTML5 and CSS3 / B. Frain. – Birmingham : Packt Publishing, 2019. – 350 с.

3. Felke-Morris T. Web Development and Design Foundations with HTML5 / T. Felke-Morris. – Boston : Pearson, 2016. – 720 с.

4. Pitt K. Making Games with JavaScript / K. Pitt. – London : O'Reilly Media, 2018. – 280 с.

5. Krug S. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability / S. Krug. – Berkeley : New Riders, 2014. – 216 с.

6. Kimberley G. Gamify Your Classroom: A Field Guide to Game-Based Learning / G. Kimberley. – San Francisco : Jossey-Bass, 2015. – 200 с.

7. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання / Національний стандарт України. – Київ : Мінекономрозвитку, 2015. – 30 с.

8. ISO/IEC 27001:2013 Information Security Management / [Електронний ресурс]. – URL: https://www.iso.org/isoiec-27001-information-security.html (дата звернення: 04.06.2025).

9. Міністерство освіти і науки України. Концепція оновлення змісту мовно-літературної освітньої галузі (іноземні мови) / МОН України. – Київ, 2025. – URL: https://mon.gov.ua/news/osvita-dlia-zhyttia-mon-rozpochynaie-onovlennia-zmistu-movno-literaturnoi-inozemni-movy-osvitnoi-haluzi (дата звернення: 04.06.2025).

10. Закон України «Про освіту» від 05.09.2017 № 2145-VIII. – Відомості Верховної Ради України, 2017, № 38-39.

11. Hamari J., Koivisto J., Sarsa H. Does Gamification Work? – A Literature Review of Empirical Studies on Gamification / J. Hamari, J. Koivisto, H. Sarsa //

Proceedings of the 47th Hawaii International Conference on System Sciences. – 2014. – P. 3025–3034.

12. Marcotte E. Responsive Web Design / E. Marcotte // A List Apart. – 2010. – URL: https://alistapart.com/article/responsive-web-design/ (дата звернення: 04.06.2025).

13. Nielsen J. Ten Usability Heuristics for User Interface Design / J. Nielsen // Nielsen Norman Group. – 1994. – URL: https://www.nngroup.com/articles/ten-usability-heuristics/ (дата звернення: 04.06.2025).

14. React Documentation / [Електронний ресурс]. – URL: https://reactjs.org/docs/getting-started.html (дата звернення: 04.06.2025).

15. Firebase Documentation / [Електронний ресурс]. – URL: https://firebase.google.com/docs (дата звернення: 04.06.2025).

16. LiqPay API Documentation / [Електронний ресурс]. – URL: https://www.liqpay.ua/documentation/en (дата звернення: 04.06.2025).

17. SEO Best Practices for Educational Websites / [Електронний ресурс]. – 2019. – URL: https://moz.com/blog/seo-for-education-websites (дата звернення: 04.06.2025).

18. Examining the effectiveness of gamification as a tool promoting teaching and learning / [Електронний ресурс]. – 2023. – URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC10591086/ (дата звернення: 04.06.2025).

19. Global Online Education Market Report 2019-2026 / [Електронний ресурс]. – URL: https://www.marketresearch.com/online-education-market-report (дата звернення: 04.06.2025).

20. Modern Web Development Technologies and Frameworks / [Електронний ресурс]. – 2021. – URL: https://www.webdevjournal.com/modern-frameworks (дата звернення: 04.06.2025).

21. UI/UX design of educational on-line courses / [Електронний ресурс]. – 2023. – URL: https://lib.iitta.gov.ua/id/eprint/733714/ (дата звернення: 04.06.2025).

22. Web quest method in online teaching Ukrainian as a foreign language / [Електронний ресурс]. – 2022. – URL: https://dialnet.unirioja.es/descarga/articulo/8271378.pdf (дата звернення: 04.06.2025).

23. Nedashkivska A., Sivachenko O. Podorozhi UA: Introductory e-textbook/course in Ukrainian for English speakers / A. Nedashkivska, O. Sivachenko. – [Електронний ресурс]. – URL: https://www.ukrainianlessons.com/ukrainian-language-resources/ (дата звернення: 04.06.2025).

24. Skillbox: сучасні онлайн-курси з іноземних мов у 2025 році / [Електронний ресурс]. – 2025. – URL: https://dtf.ru/top-raiting/3771726-luchshie-onlajn-kursy-po-inostrannym-yazykam-v-2025-godu#skillbox (дата звернення: 04.06.2025).

25. Skillfactory: гнучкі та інтерактивні курси іноземних мов / [Електронний ресурс]. – 2025. – URL: https://dtf.ru/top-raiting/3771726-luchshie-onlajn-kursy-po-inostrannym-yazykam-v-2025-godu#skillfactory (дата звернення: 04.06.2025).

26. Yandex Praktikum: адаптивне навчання іноземним мовам / [Електронний ресурс]. – 2025. – URL: https://dtf.ru/top-raiting/3771726-luchshie-onlajn-kursy-po-inostrannym-yazykam-v-2025-godu#yandex-praktikum (дата звернення: 04.06.2025).

27. UCU School of Ukrainian Language and Culture. Ukrainian Online Courses for Groups 2025 / [Електронний ресурс]. – URL: https://studyukrainian.org.ua/courses/ukrainian-online-courses-for-groups/ (дата звернення: 04.06.2025).

28. Indiana University. Ukrainian: Online Summer Language Workshop 2025 / [Електронний ресурс]. – URL: https://languageworkshop.indiana.edu/summer-language-workshop/overview/online/ukrainian/index.html (дата звернення: 04.06.2025).

29. Language International. 2025 Best Ukrainian Courses in Ukraine / [Електронний ресурс]. – URL: https://www.languageinternational.ie/ukrainian-courses-ukraine (дата звернення: 04.06.2025).

30. The Educational Equality Institute. Free Language Courses for Ukrainians / [Електронний ресурс]. – URL: https://theeducationalequalityinstitute.org/free-language-courses-for-ukrainians/ (дата звернення: 04.06.2025).

31. Lets-Learn.eu. Ukrainian Courses for Beginners & Advanced 2025 / [Електронний ресурс]. – URL: https://www.lets-learn.eu/ukrainian (дата звернення: 04.06.2025).

32. UCU Summer Program 2025 / [Електронний ресурс]. – URL: https://studyukrainian.org.ua/courses/summer-program/ (дата звернення: 04.06.2025).

33. Stony Brook University. Summer 2025 Online Courses: Ukrainian Language / [Електронний ресурс]. – URL: https://sites.utexas.edu/creees/2025/04/21/summer-2025-online-courses-russian-ukrainian-language-cinema-literature-stony-brook-university/ (дата звернення: 04.06.2025).

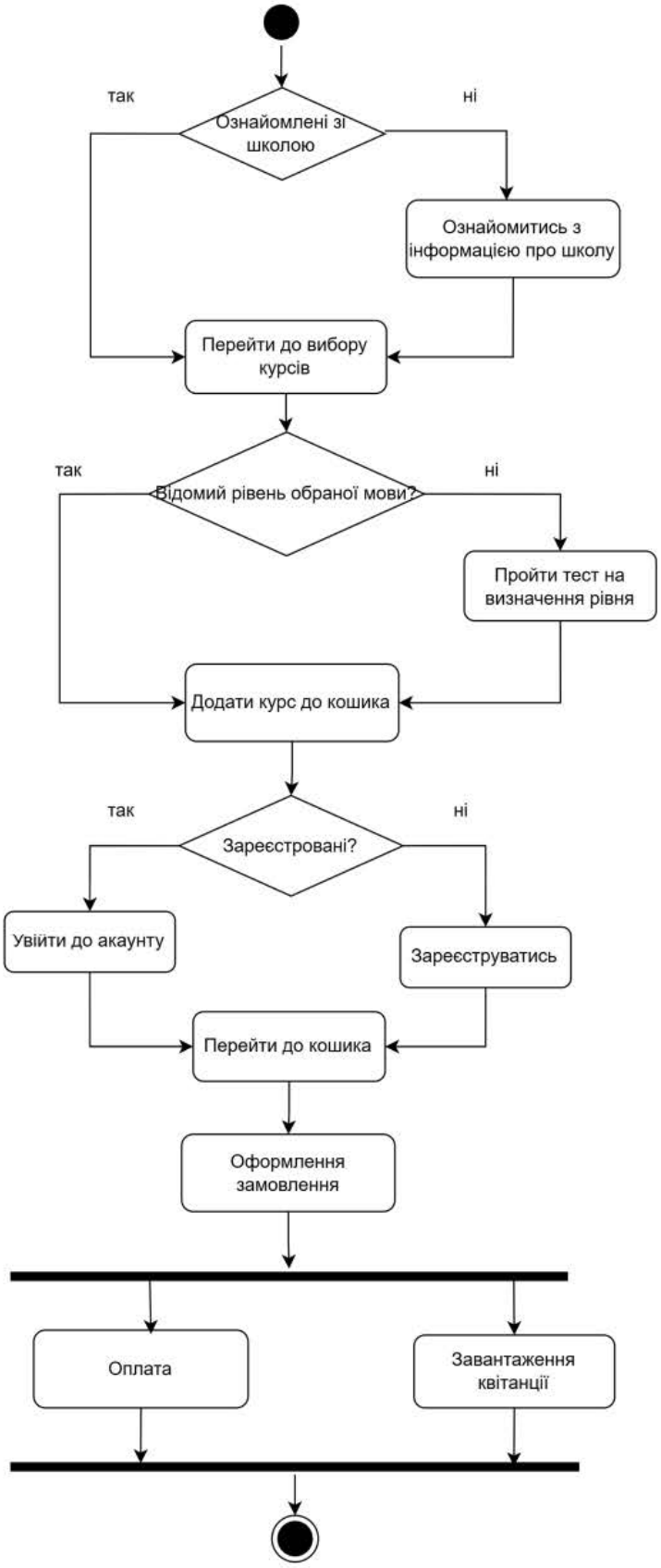
34. Udemy. Top Ukrainian Language Courses Online 2025 / [Електронний ресурс]. – URL: https://www.udemy.com/topic/ukrainian-language/ (дата звернення: 04.06.2025).

35. Український центр оцінювання якості освіти. Іноземні мови / [Електронний ресурс]. – URL: https://testportal.gov.ua/inozemni-movy-2025/ (дата звернення: 04.06.2025).

36. Ренюс. Іноземні мови в українських школах вивчатимуть по-новому: МОН анонсувало зміни у 2025 році / [Електронний ресурс]. – URL: https://renews.com.ua/kultura/inozemni-movi-v-ykrayinskih-shkolah-vivchatimyt-po-novomy-mon-anonsyvalo-zmini-y-2025-roci/ (дата звернення: 04.06.2025).

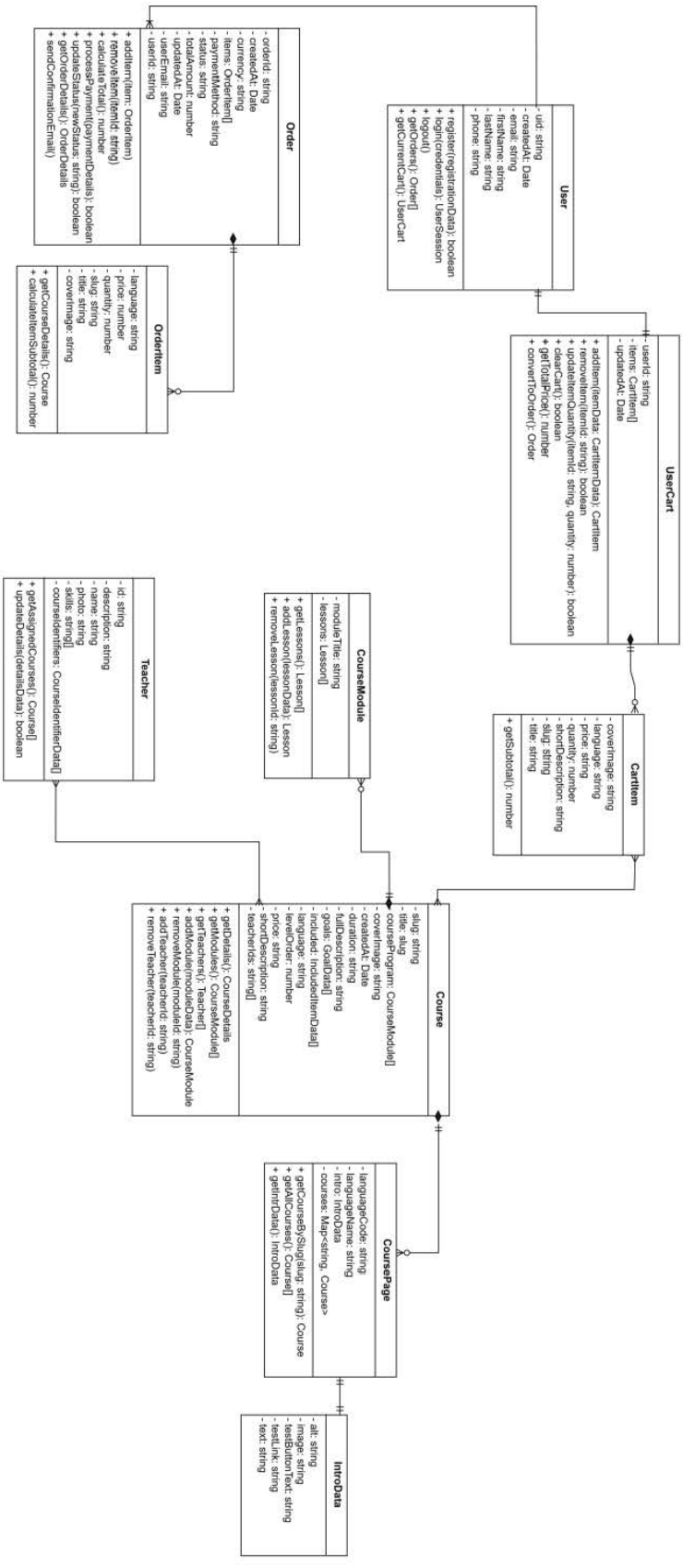
ДОДАТОК А

ДІАГРАМА АКТИВНОСТЕЙ ПРОЦЕСУ ПРИДБАННЯ КУРСУ



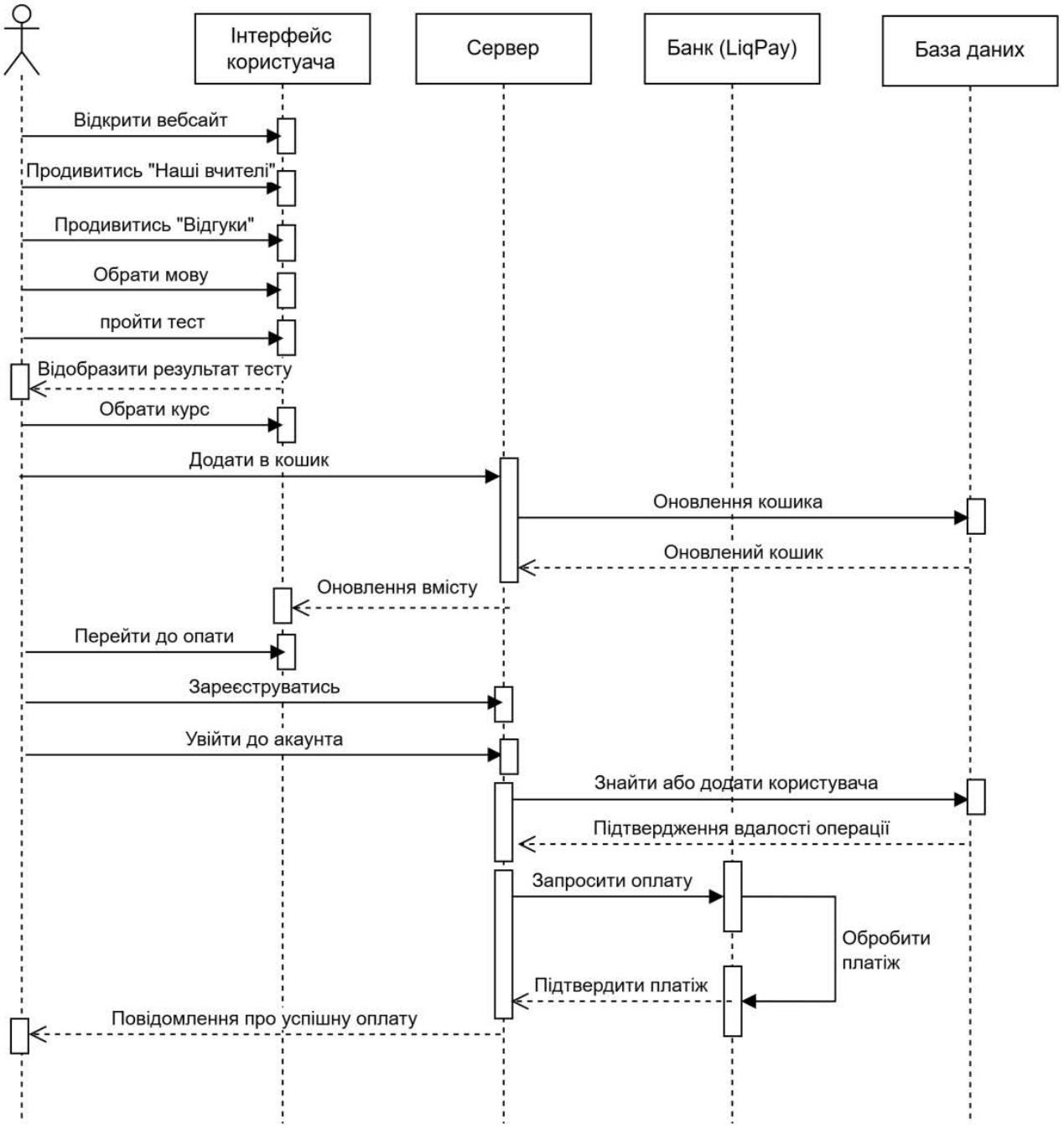
ДОДАТОК Б

ДІАГРАМА КЛАСІВ



ДОДАТОК В

ДІАГРАМА СЦЕНАРІЄВ ВЗАЄМОДІЇ З СИСТЕМОЮ



ДОДАТОК Г

ТАБЛИЦЯ АНАЛІЗУ ОЦІНОК ЯКОСТІ РОЗРОБЛЕНОГО ПРОДУКТУ

Критерій якості	Аспекти оцінювання	Методи оцінки	Результати оцінювання та висновки
Функціональність	- Повнота реалізації заявлених функцій - Коректність роботи основного та допоміжного функціоналу.	Порівняння з технічним завданням. Результати тестування.	Реалізовано всі ключові функціональні модулі. Функціонал працює коректно.
Надійність	- Стабільність роботи системи під час виконання типових сценаріїв. - Обробка помилок та виняткових ситуацій.	Результати тестування (включаючи стрес-тестування, якщо проводилося). Аналіз механізмів обробки помилок.	Система демонструє стабільну роботу. Впроваджено механізми валідації та обробки помилок, що підвищує загальну надійність.
Зручність користування (Юзабіліті)	- Легкість освоєння та виконання завдань користувачем. - Загальне задоволення від взаємодії з інтерфейсом.	Експертна оцінка. Неформальне користувацьке тестування . Аналіз логіки користувацьких шляхів.	Інтерфейс сайту оцінено як інтуїтивно зрозумілий та логічний. Користувачі можуть легко знаходити інформацію та виконувати цільові дії .
Ефективність (Швидкодія)	- Швидкість завантаження сторінок. - Час відповіді системи на дії користувача.	Інструменти аналізу швидкості (Google PageSpeed Insights, Lighthouse). Суб'єктивна оцінка часу відповіді.	Швидкість завантаження основних сторінок знаходиться в межах рекомендованих показників. Час відповіді на дії користувача є прийнятним.
Портативність (Адаптивність та Кросбраузерність)	- Коректне відображення та функціонування на різних пристроях	Тестування на різних пристроях та браузерах.	Вебсайт є повністю адаптивним та коректно працює в останніх версіях основних веббраузерів.
Безпека	- Захист персональних даних. - Безпека автентифікації та авторизації. - Захист від вебвразливостей .	Аналіз впроваджених заходів безпеки (HTTPS, хешування паролів, правила Firestore). Результати базового тестування безпеки.	Впроваджено основні заходи для забезпечення безпеки даних та автентифікації.

ДОДАТОГ Ґ.

ЛІСТИНГ КОДУ

1. Вигляд основного файлу index.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Vite + React</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

2. Вигляд файлу App.jsx

```
import { useState } from 'react'
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import './App.css'

import HomePage from './pages/HomePage';
import AboutPage from './pages/AboutPage';
import StudyingPage from './pages/StudyingPage';
import BlogPage from './pages/BlogPage';
import GamesPage from './pages/GamesPage';
import ContactsPage from './pages/ContactsPage';
import TeachersPage from './pages/TeachersPage';
import SignUpPage from './pages/SignUpPage';
import SignInPage from './pages/SignInPage';
import ReviewsPage from './pages/ReviewsPage';
import FaqPage from './pages/FaqPage';
import BlogPostPage from './pages/BlogPostPage';
import LanguageCoursePage from './pages/LanguageCoursePage';
import CourseDetailPage from './pages/CourseDetailPage/CourseDetailPage';
import CartPage from './pages/CourseDetailPage/CartPage/CartPage';
import PaymentResultPage from './pages/PaymentResultPage/PaymentResultPage.jsx';
import LiqPayCheckoutPage from './pages/LiqPayCheckoutPage/LiqPayCheckoutPage.jsx';
import JapaneseFlashcardsGame from './pages/games/JapaneseFlashcardsGame/JapaneseFlashcardsGame.jsx';
import KoreanMemoryGame from './pages/games/KoreanMemoryGame/KoreanMemoryGame.jsx';
import ChineseSentenceGame from './pages/games/ChineseSentenceGame/ChineseSentenceGame.jsx';
import EnglishQuizGame from './pages/games/EnglishQuizGame/EnglishQuizGame.jsx';
import Header from './components/layout/Header/Header';
import Footer from './components/layout/Footer/Footer';
// import Cart from './components/Cart/Cart';
function App() {
  return (
    <div className="app-container">
      <Header />
      <main className='main-content'>
        <Routes>
```

```
          <Route path='/' element={<HomePage />} />
          <Route path='/about' element={<AboutPage />} />
          <Route path='/studying' element={<StudyingPage />} />
          <Route path='/studying/:languageId' element={<LanguageCoursePage />} />
          <Route path='/courses/:languageId/:courseSlug' element={<CourseDetailPage />} />
          <Route path='/blog' element={<BlogPage />} />
          <Route path="/blog/:postSlug" element={<BlogPostPage />} />
          <Route path='/games' element={<GamesPage />} />
          <Route path="/games/japanese-flashcards" element={<JapaneseFlashcardsGame />} />
          <Route path="/games/korean-memory-match" element={<KoreanMemoryGame />} />
          <Route path="/games/chinese-sentence-builder" element={<ChineseSentenceGame />} />
           <Route path="/games/english-quiz" element={<EnglishQuizGame />} />
          <Route path='/contacts' element={<ContactsPage />} />
          <Route path='/teachers' element={<TeachersPage />} />
          <Route path='/registrate' element={<SignUpPage />} />
          <Route path='/reviews' element={<ReviewsPage />} />
          <Route path='/signin' element={<SignInPage />} />
          <Route path='/faq' element={<FaqPage />} />
          <Route path='/cart' element={<CartPage />} />
          <Route path='/payment-result' element={<PaymentResultPage />} />
          <Route path='/checkout/liqpay' element={<LiqPayCheckoutPage />} />
        </Routes>
      </main>
      <Footer />
    </div>
  )
}
export default App
```

3. Лістинг файлу BlogPage.jsx

```
import React, { useState, useMemo } from 'react';
import Header from '../components/layout/Header/Header';
import Footer from '../components/layout/Footer/Footer';
import CategoryFilter from '../components/blog/CategoryFilter/CategoryFilter';
import BlogCard from '../components/blog/BlogCard/BlogCard';
import Pagination from '../components/blog/Pagination/Pagination';
import blogPostsData, { blogCategories } from '../data/blogPostsData';
import './BlogPage.css';

const POSTS_PER_PAGE = 3;
function BlogPage() {
  const [selectedCategory, setSelectedCategory] = useState('All');
  const [currentPage, setCurrentPage] = useState(1);
  const filteredPosts = useMemo(() => {
    if (selectedCategory === 'All') {
      return blogPostsData;
    }
    return blogPostsData.filter(post => post.category === selectedCategory);
  }, [selectedCategory]);
  const totalPages = Math.ceil(filteredPosts.length / POSTS_PER_PAGE);
  const displayedPosts = useMemo(() => {
    const startIndex = (currentPage - 1) * POSTS_PER_PAGE;
    const endIndex = startIndex + POSTS_PER_PAGE;
    return filteredPosts.slice(startIndex, endIndex);
  }, [filteredPosts, currentPage]);
  const handleCategoryChange = (category) => {
    setSelectedCategory(category);
    setCurrentPage(1);
  };
  const handlePageChange = (pageNumber) => {
```

```
    setCurrentPage(pageNumber);
  };
  return (
    <>
      <main className="blog-page-main">
        <div className="container">
          <h1 className="blog-title">Blog</h1>
          <CategoryFilter
            categories={blogCategories}
            selectedCategory={selectedCategory}
            onSelectCategory={handleCategoryChange}
          />
          {displayedPosts.length > 0 ? (
            <div className="blog-grid">
              {displayedPosts.map((post) => (
                <BlogCard
                  key={post.id}
                  id={post.id}
                  slug={post.slug}
                  title={post.cardTitle || post.title}
                  description={post.description}
                  image={post.image}
                />
              ))}
            </div>
          ) : (
            <p className="no-posts-message">Немає статей у цій категорії.</p>
          )}
          {totalPages > 1 && (
            <Pagination
              currentPage={currentPage}
              totalPages={totalPages}
              onPageChange={handlePageChange}
            />
          )}
        </div>
      </main>
    </>
  );
}
export default BlogPage;
```

4. Лістинг файлу LanguageCoursePage.jsx

```
import React, { useState, useEffect } from "react";
import { useParams, Link } from "react-router-dom";
import { doc, getDoc } from "firebase/firestore";
import { db } from "../firebase";
import './LanguageCoursePage.css';

import CourseIntro from "../components/language/CourseIntro/CourseIntro.jsx";
import CourseSection from "../components/language/CourseSection/CourseSection.jsx";
import Loader from "../components/common/Loader/Loader.jsx";
function LanguageCoursePage() {
    const { languageId } = useParams();
    const [languageData, setLanguageData] = useState(null);
    const [loading, setLoading] = useState(true);
    const [error, setError] = useState(null);
    useEffect(() => {
        const fetchLanguageData = async () => {
            setLoading(true);
```

```
            setError(null);
            try {
                const docRef = doc(db, "courses", languageId);
                const docSnap = await getDoc(docRef);
                if (docSnap.exists()) {
                    setLanguageData(docSnap.data());
                } else {
                    setError(`Дані для мови "${languageId}" не знайдено`);
                }
            } catch (err) {
                console.error("Помилка завантаження даних мови:", err);
                setError("Помилка завантаження даних");
            } finally {
                setLoading(false);
            }
        };
        fetchLanguageData();
    }, [languageId]);
    if (loading) {
        return <Loader />;
    }
    if (error) {
        return (
            <main className="language-course-page-main error-page">
                <div className="container">
                    <h1>{error}</h1>
                    <p>На жаль, ми не змогли завантажити інформацію.</p>
                    <Link to="/studying" className="back-link">← Повернутись до вибору мови</Link>
                </div>
            </main>
        );
    }
    if (!languageData) {
        return null;
    }
    const { languageName, intro, courses } = languageData;
    console.log("Language Data from Firestore:", languageData);
    console.log("Intro object from languageData:", intro);
    if (intro) {
        console.log("intro.testLink:", intro.testLink);
        console.log("intro.testButtonText:", intro.testButtonText);
    }
    const courseList = courses ? Object.values(courses) : [];
    if (courseList.length > 0 && courseList[0].hasOwnProperty('levelOrder')) {
        courseList.sort((a, b) => (a.levelOrder || 0) - (b.levelOrder || 0));
    }
    return (
        <main className="language-course-page-main">
            <div className="container">
                <Link to="/studying" className="back-link">← Назад до вибору мови</Link>
                {intro && (
                    <CourseIntro
                        languageName={languageName}
                        text={intro.text}
                        image={intro.image}
                        alt={intro.alt}
                        buttonText={intro.testButtonText}
                        testLink={intro.testLink}
                    />
                )}
                {courseList.length > 0 && courseList.map((course, index) => (
                    <React.Fragment key={course.slug || index}>
                        <hr className="section-divider" />
```

```
                        <CourseSection
                            course={course}
                            languageId={languageId}
                            layout={index % 2 === 0 ? 'image-left' : 'image-right'}
                        />
                    </React.Fragment>
                ))}
                {courseList.length === 0 && <p>Для цієї мови ще немає доступних курсів.</p>}
            </div>
        </main>
    );
}
export default LanguageCoursePage;
```

5. Лістинг файлу LiqPayService.js

```
export const createPayment = async (orderData) => {
    console.log("[Client liqpayService] createPayment called with orderData:", orderData);
    try {
        const preparePaymentFunctionUrl = "https://prepareliqpaypaymentdata-mhmryvx76q-ew.a.run.app";

        const response = await fetch(preparePaymentFunctionUrl, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({
                orderId: orderData.orderId,
                amount: orderData.amount,
                description: orderData.description,
            }),
        });
        if (!response.ok) {
            const errorData = await response.json().catch(() => ({ error: "Unknown server error or non-
JSON response" }));
            console.error("[Client liqpayService] Error from prepareLiqPayPaymentData function:",
response.status, errorData);
            throw new Error(errorData.error || `Server error: ${response.status}`);
        }
        const paymentFormParams = await response.json();
        console.log("[Client liqpayService] Received form params from backend:", paymentFormParams);
        if (!paymentFormParams || !paymentFormParams.data || !paymentFormParams.signature) {
            console.error("[Client liqPayService] Invalid data/signature received from backend.",
paymentFormParams);
            throw new Error("Неможливо отримати коректні параметри для форми LiqPay від сервера.");
        }
        return {
            data: paymentFormParams.data,
            signature: paymentFormParams.signature,
        };
    } catch (error) {
        console.error('[Client liqpayService] Error in createPayment while fetching from backend:', error);
        if (error.message.includes("Failed to fetch")) {
            throw new Error("Помилка мережі під час звернення до сервера для підготовки платежу. Перевірте
ваше інтернет-з'єднання.");
        }
        throw error;
    }
};
export const getPaymentStatus = async (orderId) => {
    console.log("[Client liqpayService] getPaymentStatus called for orderId:", orderId);
```

```
    if (!orderId) {
        console.error("[Client liqpayService] orderId is required for getPaymentStatus.");
        throw new Error("orderId is required to get payment status.");
    }
    try {
        const getStatusFunctionUrl = "https://getliqpayorderstatus-mhmryvx76q-ew.a.run.app";
        const response = await fetch(`${getStatusFunctionUrl}?orderId=${encodeURIComponent(orderId)}`, {
            method: 'GET',
            headers: {
                'Content-Type': 'application/json',
            },
        });
        const responseData = await response.json();
        if (!response.ok) {
            console.error("[Client liqPayService] Error from getLiqPayOrderStatus function:",
response.status, responseData);
            throw new Error(responseData.error || responseData.details || `Server error while fetching
status: ${response.status}`);
        }
        console.log("[Client liqpayService] Received payment status from backend:", responseData);
        return responseData;
    } catch (error) {
        console.error('[Client liqpayService] Error in getPaymentStatus:', error);
        if (error.message.includes("Failed to fetch")) {
            throw new Error("Помилка мережі під час запиту статусу платежу. Перевірте інтернет-
з'єднання.");
        }
        throw error;
    }
};
```

6. Лістинг файлу CartContext.jsx

```
import React, { createContext, useState, useContext, useEffect, useCallback } from "react";
import { useAuth } from "./AuthContext";
import { db } from "../firebase";
import { doc, getDoc, setDoc, serverTimestamp } from 'firebase/firestore';

const CartContext = createContext();
export const useCart = () => useContext(CartContext);
const LOCAL_STORAGE_CART_KEY = 'anonymousCartItems';
export const CartProvider = ({ children }) => {
    const [cartItems, setCartItems] = useState([]);
    const [loadingCart, setLoadingCart] = useState(true);
    const { currentUser, loadingAuthState } = useAuth();
    const saveCart = useCallback(async (uid, items) => {
        if (!uid) {
            localStorage.setItem(LOCAL_STORAGE_CART_KEY, JSON.stringify(items));
            return;
        }

        try {
            await setDoc(doc(db, 'userCarts', uid), {
                items,
                updatedAt: serverTimestamp()
            });
        } catch (error) {
            console.error("Ошибка сохранения корзины:", error);
        }
    }, []);
    const loadCart = useCallback(async () => {
        setLoadingCart(true);
        try {
```

```
            if (currentUser) {
                const docSnap = await getDoc(doc(db, 'userCarts', currentUser.uid));
                if (docSnap.exists()) {
                    setCartItems(docSnap.data().items || []);
                } else {
                    await saveCart(currentUser.uid, []);
                }
            } else {
                const localCart = localStorage.getItem(LOCAL_STORAGE_CART_KEY);
                setCartItems(localCart ? JSON.parse(localCart) : []);
            }
        } catch (error) {
            console.error("Ошибка загрузки корзины:", error);
            setCartItems([]);
        } finally {
            setLoadingCart(false);
        }
    }, [currentUser, saveCart]);
    useEffect(() => {
        if (!loadingAuthState) {
            loadCart();
        }
    }, [loadingAuthState, loadCart]);
    useEffect(() => {
        if (loadingCart || loadingAuthState) return;
         const timer = setTimeout(() => {
            saveCart(currentUser?.uid, cartItems);
        }, 500);
        return () => clearTimeout(timer);
    }, [cartItems, currentUser, loadingCart, loadingAuthState, saveCart]);
    const addItemToCart = useCallback((product) => {
        setCartItems(prev => {
            const existing = prev.find(item =>
                item.slug === product.slug &&
                item.language === product.language
            );

            return existing
                ? prev.map(item =>
                    item.slug === product.slug && item.language === product.language
                        ? { ...item, quantity: item.quantity + 1 }
                        : item
                  )
                : [...prev, { ...product, quantity: 1 }];
        });
    }, []);
    const removeItemFromCart = useCallback((slug, language) => {
        setCartItems(prev => prev.filter(item =>
            !(item.slug === slug && item.language === language)
        ));
    }, []);
    const updateItemQuantity = useCallback((slug, language, quantity) => {
        setCartItems(prev => prev.map(item =>
            item.slug === slug && item.language === language
                ? { ...item, quantity: Math.max(1, quantity) }
                : item
        ).filter(item => item.quantity > 0));
    }, []);
   const clearCart = useCallback(async () => {
        setCartItems([]);
        if (currentUser) {
            await saveCart(currentUser.uid, []);
        } else {
```

```
            localStorage.removeItem(LOCAL_STORAGE_CART_KEY);
        }
    }, [currentUser, saveCart]);
    const getCartTotal = () => cartItems.reduce(
        (total, item) => total + (parseFloat(item.price) * item.quantity), 0
    );
    const getTotalItems = () => cartItems.reduce(
        (total, item) => total + item.quantity, 0
    );
    return (
        <CartContext.Provider value={{
            cartItems,
            loadingCart,
            addItemToCart,
            removeItemFromCart,
            updateItemQuantity,
            clearCart,
            getTotalItems: () => cartItems.reduce((total, item) => total + item.quantity, 0),
            getCartTotal: () => cartItems.reduce((total, item) =>
                total + (parseFloat(item.price) * item.quantity), 0)
        }}>
            {children}
        </CartContext.Provider>
    );
};
```

7. Лістинг коду SignUpForm.jsx

```
import React, {useState} from "react";
import { useNavigate } from "react-router-dom";
import { createUserWithEmailAndPassword, updateProfile } from "firebase/auth";
import { doc, setDoc } from 'firebase/firestore';
import { auth, db } from '../../../firebase.js';
import InputField from "../../common/InputField/InputField.jsx";
import FormBtn from "../FormBtn/FormBtn.jsx";
import './SignUpForm.css';

import UserIcon from '../../../assets/icons/UserIcon.svg';
import EnvelopeIcon from '../../../assets/icons/EnvelopeIcon.svg';
import PhoneIcon from '../../../assets/icons/PhoneIcon.svg';
import KeyIcon from '../../../assets/icons/KeyIcon.svg';
const SignUpForm = () => {
    const [formData, setFormData] = useState({
        firstName: '', lastName: '', email: '', phone: '', password: '', confirmPassword: ''
    });
    const [errors, setErrors] = useState({});
    const [loading, setLoading] = useState(false);
    const [firebaseError, setFirebaseError] = useState('');
    const navigate = useNavigate();
    const handleChange = (field) => (e) => {
        setFormData(prev => ({ ...prev, [field]: e.target.value }));
        if (errors[field]) setErrors(prev => ({ ...prev, [field]: '' }));
        if (firebaseError) setFirebaseError('');
    };
    const validateForm = () => {
        const newErrors = {};
        if (!formData.firstName) newErrors.firstName = 'First name is required';
        if (!formData.lastName) newErrors.lastName = 'Last name is required';
        if (!formData.email) {
            newErrors.email = 'Email is required';
        } else if (!/\S+@\S+\.\S+/.test(formData.email)) {
```

```
            newErrors.email = 'Email is invalid';
        }
        if (!formData.phone) newErrors.phone = 'Phone number is required';
        if (!formData.password) {
            newErrors.password = 'Password is required';
        } else if (formData.password.length < 6) {
            newErrors.password = 'Password must be at least 6 characters';
        }
        if (formData.password !== formData.confirmPassword) {
            newErrors.confirmPassword = 'Passwords do not match';
        }
        setErrors(newErrors);
        return Object.keys(newErrors).length === 0;
    };
    const handleSubmit = async (e) => {
        e.preventDefault();
        setFirebaseError('');
        if (!validateForm()) return;
        setLoading(true);
        try {
            const userCredential = await createUserWithEmailAndPassword(
                auth,
                formData.email,
                formData.password
            );
            const user = userCredential.user;
            await updateProfile(user, {
                displayName: `${formData.firstName} ${formData.lastName}`
            });
            await setDoc(doc(db, "users", user.uid), {
                uid: user.uid,
                firstName: formData.firstName,
                lastName: formData.lastName,
                email: formData.email,
                phone: formData.phone,
                createdAt: new Date()
            });
            console.log('User registered successfully:', user);
            navigate('/');
        } catch (error) {
            console.error('Error during registration:', error);
            if (error.code === 'auth/email-already-in-use') {
                setFirebaseError('This email address is already in use.');
            } else if (error.code === 'auth/weak-password') {
                setFirebaseError('Password is too weak. Must be at least 6 characters.');
            } else {
                setFirebaseError('Failed to register. Please try again.');
            }
        } finally {
            setLoading(false);
        }
    };
    return (
        <form className="signup-form" onSubmit={handleSubmit}>
            {firebaseError && <p className="firebase-error-message">{firebaseError}</p>}
            <div className="form-row">
                <InputField label="First Name" icon={UserIcon} placeholder="Enter your first name"
value={formData.firstName} onChange={handleChange('firstName')} error={errors.firstName} required
disabled={loading} />
                <InputField label="Last Name" icon={UserIcon} placeholder="Enter your last name"
value={formData.lastName} onChange={handleChange('lastName')} error={errors.lastName} required
disabled={loading} />
            </div>
```

```
            <InputField label="Email" type="email" icon={EnvelopeIcon} placeholder="Enter your email"
value={formData.email} onChange={handleChange('email')} error={errors.email} required disabled={loading}
/>
            <InputField label="Phone Number" type="tel" icon={PhoneIcon} placeholder="Enter your phone
number" value={formData.phone} onChange={handleChange('phone')} error={errors.phone} required
disabled={loading} />
            <InputField label="Password" type="password" icon={KeyIcon} placeholder="Enter your password"
value={formData.password} onChange={handleChange('password')} error={errors.password} required
disabled={loading} />
            <InputField label="Confirm Password" type="password" icon={KeyIcon} placeholder="Confirm your
password" value={formData.confirmPassword} onChange={handleChange('confirmPassword')}
error={errors.confirmPassword} required disabled={loading} />
            <FormBtn btnValue={loading ? 'Registering...' : 'Register'} disabled={loading} />
        </form>
    );
};
export default SignUpForm;
```

8. Лістинг файлу SignInForm.jsx

```
import React, {useState} from "react";
import { useNavigate } from "react-router-dom";
import { signInWithEmailAndPassword } from "firebase/auth";
import { auth } from '../../../firebase';
import InputField from "../../common/InputField/InputField";
import FormBtn from "../FormBtn/FormBtn";
import '../SignUpForm/SignUpForm.css';

import EnvelopeIcon from '../../../assets/icons/EnvelopeIcon.svg';
import KeyIcon from '../../../assets/icons/KeyIcon.svg';
const SignInForm = () => {
    const [formData, setFormData] = useState({ email: '', password: '' });
    const [error, setError] = useState('');
    const [loading, setLoading] = useState(false);
    const navigate = useNavigate();
    const handleChange = (field) => (e) => {
        setFormData(prev => ({ ...prev, [field]: e.target.value }));
        if (error) setError('');
    };
    const handleSubmit = async (e) => {
        e.preventDefault();
        setError('');
        if (!formData.email || !formData.password) {
            setError('Please enter both email and password.');
            return;
        }
        setLoading(true);
        try {
            await signInWithEmailAndPassword(auth, formData.email, formData.password);
            console.log('User signed in successfully');
            navigate('/');
        } catch (err) {
            console.error('Error signing in:', err);
            if (err.code === 'auth/user-not-found' || err.code === 'auth/wrong-password' || err.code ===
'auth/invalid-credential') {
                setError('Invalid email or password.');
            } else {
                setError('Failed to sign in. Please try again.');
            }
        } finally {
            setLoading(false);
```

```
        }
    };
    return (
        <form className="signin-form" onSubmit={handleSubmit}>
            {error && <p className="firebase-error-message">{error}</p>}
            <InputField label="Email" type="email" icon={EnvelopeIcon} placeholder="Enter your email"
value={formData.email} onChange={handleChange('email')} required disabled={loading} />
            <InputField label="Password" type="password" icon={KeyIcon} placeholder="Enter your password"
value={formData.password} onChange={handleChange('password')} required disabled={loading} />
            <FormBtn btnValue={loading ? 'Signing In...' : 'Sign In'} disabled={loading} />
        </form>
    );
};
export default SignInForm;
```

9. Лістинг файлу функцій для системи LiqPay index.js

```
import { onRequest } from "firebase-functions/v2/https";
import { setGlobalOptions } from "firebase-functions/v2";
import admin from "firebase-admin";
import crypto from "crypto";


import { createRequire } from "node:module";
const require = createRequire(import.meta.url);
const LiqPay = require("liqpay");
setGlobalOptions({ region: "europe-west1" });
if (!admin.apps.length) {
  admin.initializeApp();
}
const db = admin.firestore();
function generateLiqPaySignature(privateKey, data) {
  const signString = String(privateKey) + String(data) + String(privateKey);
  return crypto.createHash("sha1").update(signString).digest("base64");
}
export const liqpayCallback = onRequest(
  { region: "europe-west1", cors: true },
  async (req, res) => {
    console.log("liqpayCallback: Invoked (HARDCODED KEYS TEST). Method:", req.method);
    if (req.method !== "POST") {
      console.warn("liqpayCallback: Received non-POST request.");
      return res.status(405).send("Method Not Allowed");
    }
    console.log("liqpayCallback: Request body (data part - first 100 chars):", {
      data: String(req.body.data).substring(0,100) + (String(req.body.data).length > 100 ? "..." : ""),
      signature: req.body.signature
    });
    try {
      const receivedData = req.body.data;
      const receivedSignature = req.body.signature;
      const PRIVATE_KEY = HARDCODED_PRIVATE_KEY;
      if (!PRIVATE_KEY) {
        console.error("liqpayCallback: (Hardcoded) Private key is missing!");
        return res.status(500).send("Internal Server Error: LiqPay private key misconfigured.");
      }
      if (!receivedData || !receivedSignature) {
        console.warn("liqpayCallback: Missing data/signature in POST body.");
        return res.status(400).send("Bad Request: Missing data/signature.");
      }
      const calculatedSignature = generateLiqPaySignature(PRIVATE_KEY, receivedData);
      if (calculatedSignature !== receivedSignature) {
```

```
        console.error(`liqpayCallback: Invalid signature. Expected: ${calculatedSignature}, Received:
${receivedSignature}.`);
        return res.status(400).send("Invalid signature");
      }
      console.log("liqpayCallback: Signature VERIFIED.");
      let paymentDetails;
      try {
        const jsonString = Buffer.from(receivedData, "base64").toString("utf-8");
        paymentDetails = JSON.parse(jsonString);
        console.log("liqpayCallback: Payment details decoded:", paymentDetails);
      } catch (error) {
        console.error("liqpayCallback: Error decoding data from LiqPay callback.", error);
        return res.status(400).send("Error decoding data");
      }
      const orderId = paymentDetails.order_id;
      if (!orderId || typeof orderId !== "string") {
        console.error("liqpayCallback: Invalid order_id in payment details:", paymentDetails);
        return res.status(400).send("Bad Request: Invalid order_id");
      }
      const status = paymentDetails.status;
      const paymentId = paymentDetails.payment_id || "N/A";
      console.log(`liqpayCallback: Updating Firestore for order_id=${orderId}, status=${status},
payment_id=${paymentId}`);
      const orderRef = db.collection("orders").doc(orderId);
      let newOrderStatus;
      let updatePayload = {
        liqpayPaymentId: paymentId,
        liqpayStatus: status,
        liqpayCallbackDataRaw: receivedData,
        updatedAt: admin.firestore.FieldValue.serverTimestamp(),
      };
      if (status === "success" || status === "sandbox") { newOrderStatus = "paid"; }
      else if (status === "failure" || status === "error" || status === "reversed") {
        newOrderStatus = "payment_failed";
        updatePayload.liqpayErrorDescription = paymentDetails.err_description || paymentDetails.err_code
|| "Unknown LiqPay error";
      } else { newOrderStatus = `payment_processing_${status}`; }
      updatePayload.status = newOrderStatus;
      const docSnap = await orderRef.get();
      if (docSnap.exists) {
        await orderRef.update(updatePayload);
        console.log(`liqpayCallback: Firestore updated for order ${orderId} with status
${newOrderStatus}.`);
      } else {
        console.warn(`liqpayCallback: Order ${orderId} not found. Creating record in
'liqpay_orphaned_callbacks'.`);
        await db.collection("liqpay_orphaned_callbacks").doc(orderId).set({ ...paymentDetails,
processedSignature: calculatedSignature, receivedSignature: receivedSignature, receivedAt:
admin.firestore.FieldValue.serverTimestamp(), note: "Order document not found at time of callback."});
      }
      return res.status(200).send("OK");
    } catch (error) {
      console.error("liqpayCallback: Unhandled error:", error);
      if (error instanceof Error) { console.error("Error stack:", error.stack); }
      return res.status(500).send("Internal Server Error");
    }
  }
);
export const prepareLiqPayPaymentData = onRequest(
  { cors: true, region: "europe-west1" },
  async (req, res) => {
    console.log("--- prepareLiqPayPaymentData Invoked (Manual Data/Signature - Checkpoint C) ---");
    if (req.method !== "POST") {
```

```
      console.warn("prepareLiqPayPaymentData: Received non-POST request.");
      return res.status(405).json({ error: "Method Not Allowed" });
    }
    try {
      const { orderId, amount, description } = req.body;
      console.log("prepareLiqPayPaymentData: Request body:", req.body);
      if (!orderId || typeof amount === "undefined" || !description || typeof amount !== "number" ||
amount <= 0) {
        console.error("prepareLiqPayPaymentData: Invalid or missing parameters.", req.body);
        return res.status(400).json({ error: "Invalid or missing parameters." });
      }
      const PUBLIC_KEY = HARDCODED_PUBLIC_KEY;
      const PRIVATE_KEY = HARDCODED_PRIVATE_KEY;
      const SANDBOX_MODE = HARDCODED_SANDBOX_MODE;
      const RESULT_URL = HARDCODED_RESULT_URL;
      const SERVER_URL_CALLBACK = HARDCODED_SERVER_URL_CALLBACK;
      console.log("prepareLiqPayPaymentData: Using HARDCODED LiqPay params for data/signature
generation.");
      const liqpayParams = {
        public_key: PUBLIC_KEY,
        action: "pay",
        amount: Number(amount),
        currency: "UAH",
        description: description,
        order_id: orderId,
        version: "3",
        sandbox: SANDBOX_MODE,
        result_url: RESULT_URL,
        server_url: SERVER_URL_CALLBACK,
        language: "uk",
      };
      console.log("prepareLiqPayPaymentData: Params for LiqPay (before base64):", liqpayParams);
      const jsonData = JSON.stringify(liqpayParams);
      const dataBase64 = Buffer.from(jsonData).toString("base64");
      console.log("prepareLiqPayPaymentData: Generated data (base64 - first 50):", dataBase64.substring(0,
50) + "...");
      const signatureBase64 = generateLiqPaySignature(PRIVATE_KEY, dataBase64);
      console.log("prepareLiqPayPaymentData: Generated signature (first 50):",
signatureBase64.substring(0,50) + "...");
      return res.status(200).json({
        data: dataBase64,
        signature: signatureBase64,
      });
    } catch (error) {
      console.error("prepareLiqPayPaymentData: Unhandled error (Manual Data/Signature):", error);
      const msg = error instanceof Error ? error.message : "Unknown error.";
      if (error instanceof Error) { console.error("Stack:", error.stack); }
      return res.status(500).json({ error: "Internal error preparing payment data (manual).", details:
msg });
    }
  }
);
export const getLiqPayOrderStatus = onRequest(
  { cors: true, region: "europe-west1" },
  async (req, res) => {
    console.log("getLiqPayOrderStatus: Invoked (HARDCODED KEYS TEST). Method:", req.method);
    if (req.method !== "GET") {
      console.warn("getLiqPayOrderStatus: Received non-GET request.");
      return res.status(405).json({ error: "Method Not Allowed" });
    }
    const orderId = req.query.orderId;
    console.log("getLiqPayOrderStatus: Request query:", req.query);
    if (!orderId) {
```

```
      console.error("getLiqPayOrderStatus: Missing orderId in query parameters.");
      return res.status(400).json({ error: "Missing orderId parameter." });
    }
    try {
      const PUBLIC_KEY = HARDCODED_PUBLIC_KEY;
      const PRIVATE_KEY = HARDCODED_PRIVATE_KEY;
      const liqpay = new LiqPay(PUBLIC_KEY, PRIVATE_KEY);
      console.log("[getLiqPayOrderStatus SDK Test] Type of liqpay object:", typeof liqpay);
      console.log("[getLiqPayOrderStatus SDK Test] Does liqpay.api exist?", typeof liqpay.api);
      const paramsForStatus = {
        action: "status",
        version: "3",
        order_id: orderId,
      };
      console.log("getLiqPayOrderStatus: Params for LiqPay API status request:", paramsForStatus);
      if (typeof liqpay.api !== "function") {
        console.error("getLiqPayOrderStatus: liqpay.api IS NOT A FUNCTION! Check LiqPay SDK.");
        throw new Error("LiqPay SDK's api method is not available.");
      }
      const paymentStatus = await new Promise((resolve, reject) => {
        liqpay.api("request", paramsForStatus,
          (json) => {
            console.log("getLiqPayOrderStatus: LiqPay API response (success):", json);
            resolve(json);
          },
          (errLiqpay, dataErrLiqpay) => {
            console.error("getLiqPayOrderStatus: LiqPay API error:", errLiqpay, dataErrLiqpay);
            const errorResponse = dataErrLiqpay || errLiqpay || { error: "Failed to get payment status
from LiqPay" };
            reject(errorResponse);
          });
      });
      if (paymentStatus.result === "error" || paymentStatus.status === "error") {
        console.warn("getLiqPayOrderStatus: LiqPay returned error status for orderId:", orderId,
paymentStatus);
        return res.status(400).json({
          error: "LiqPay returned an error for this order.",
          details: paymentStatus,
        });
      }
      console.log("getLiqPayOrderStatus: Successfully fetched status for orderId:", orderId,
paymentStatus);
      return res.status(200).json(paymentStatus);
    } catch (error) {
      console.error("getLiqPayOrderStatus: Unhandled error (HARDCODED KEYS TEST):", error);
      const errorMessage = error instanceof Error ? error.message : "Unknown internal server error.";
      if (error instanceof Error) { console.error("Error stack:", error.stack); }
      if (typeof error === "object" && error !== null && error.error) {
        return res.status(500).json(error);
      }
      return res.status(500).json({ error: "Internal server error while fetching payment status (hardcoded
test).", details: errorMessage });
    }
  }
);
```