МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
Кафедра програмної інженерії

КОМПЛЕКСНИЙ КУРСОВИЙ ПРОЄКТ
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)
Інтерактивний веб-додаток для навчання англійської мови із чат-ботом на базі AI

Виконав:
здобувач (ка) 3 курсу, групи ПЗПІ-22-9
Катерина М'ЯЧ

Спеціальність 121 – Інженерія програмного забезпечення
Тип програми освітньо-професійна
Освітня програма Програмна інженерія

Керівний ст.викл. кафедри ПІ Катерина ЗИБІНА

Члени комісії
Доц. ВЕЧУР О.В.
Доц. КРАВЕЦЬ Н.С.
Доц. РАБОТЯГОВ А.В.

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія

Курс 3 Група ПЗПІ-22-9 Семестр 6

ЗАВДАННЯ

на курсовий проект(роботу) студента

здобувачеві М’яч Катерини Олександрівни

1. Тема роботи Інтерактивний веб-додаток для навчання англійської мови із чат-ботом на базі AI
2. Термін здачі студентом закінченої роботи „ 20 ” червня 2025 р.
3. Вихідні дані до проєкту Методичні вказівки до виконання курсової роботи, вимоги до інформаційної системи, предметна область, що пов’язана з інтерактивним вивченням іноземних мов за допомогою штучного інтелекту
4. Перелік питань, що потрібно опрацювати в роботі

Провести системний аналіз, розробити архітектуру та базу даних, реалізувати основний функціонал клієнтської та серверної частин веб-додатку, провести тестування та проаналізувати отримані результати

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	17.03.25 – 25.03.25	виконано
2	Розробка постановки задачі	30.03.25 – 02.04.25	виконано
3	Проектування ПЗ	02.04.25 – 15.04.25	виконано
4	Програмна реалізація	15.04.25 – 01.05.25	виконано
5	Аналіз результатів	01.05.25 – 16.05.25	виконано
6	Підготовка пояснювальної записки	01.06.25 – 17.06.25	виконано
7	Перевірка на наявність ознак академічного плагіату	18.06.25 – 20.06.25	виконано
8	Захист роботи	З 09.06.25	виконано

Дата видачі завдання “ 26 ” лютого 2025 р.

Здобувач _____

Керівник роботи _____ ст.викл. кафедри ПІ Катерина ЗИБІНА

РЕФЕРАТ / ABSTRACT

Пояснювальна записка містить: 62 с., 16 рис., 3 табл., 10 джерел.

ВИВЧЕННЯ МОВ, ШТУЧНИЙ ІНТЕЛЕКТ, ВЕБ-ЗАСТОСУНОК, ЧАТ-БОТ, ІНТЕРАКТИВНІ ВПРАВИ, NODE.JS, REACT, SEQUELIZE, SQLITE, GOOGLE GEMINI

Об’єктом розробки є інтерактивний веб-додаток для навчання англійської мови з чат-ботом на базі AI.

Метою розробки є проєктування та реалізація програмної системи, що надає користувачам персоналізований досвід вивчення мови через адаптивні вправи та листування з AI-асистентом. Система спрямована на покращення словникового запасу та граматичних навичок.

Методи розробки програмної системи для серверної частини базуються на платформі Node.js з використанням фреймворку Express для створення REST API та ORM Sequelize для взаємодії з базою даних SQLite. Клієнтська частина створена за допомогою бібліотеки React для побудови динамічного користувацького інтерфейсу. Інтерактивний чат-бот реалізовано шляхом інтеграції з генеративним штучним інтелектом через Google Gemini API.

У результаті розробки спроєктовано та розроблено веб-додаток для вивчення англійської мови, який дозволяє користувачам проходити вправи відповідно до їхнього рівня знань, переписуватися з AI-асистентом на обрані теми та вести персональний словник.

LANGUAGE LEARNING, ARTIFICIAL INTELLIGENCE, WEB APPLICATION, CHATBOT, INTERACTIVE EXERCISES, NODE.JS, REACT, SEQUELIZE, SQLITE, GOOGLE GEMINI

The object of development is an interactive web application for learning the English language with an AI-based chatbot.

The goal of the development is to design and implement a software system that provides users with a personalized language learning experience through adaptive exercises and text-based correspondence with an AI assistant. The system is aimed at improving vocabulary and grammar skills.

The development methods for the server-side of the software system are based on the Node.js platform using the Express framework to create a REST API and the Sequelize ORM for interaction with the SQLite database. The client side was created using the React library to build a dynamic user interface. The interactive chatbot is implemented through integration with generative artificial intelligence via the Google Gemini API.

As a result of the development, a web application for learning English has been designed and developed, which allows users to complete exercises according to their knowledge level, correspond with an AI assistant on selected topics, and maintain a personal dictionary.

ЗМІСТ

ВСТУП

На сучасному етапі глобалізації володіння англійською мовою перетворилося з переваги на фундаментальну необхідність для успішної професійної діяльності, академічного розвитку та міжкультурної комунікації. Попит на ефективні та доступні інструменти для вивчення мов стрімко зростає. Однак традиційні методи, такі як групові заняття або робота з підручниками, часто не враховують індивідуальний темп та потреби учня, а персональні заняття з репетитором є фінансово затратними для широкого кола людей. Крім того, однією з ключових проблем у вивченні мови є відсутність регулярної практики письмової комунікації, що є критично важливим для подолання мовного бар'єру [9].

Цифровізація освіти та розвиток технологій штучного інтелекту відкривають нові горизонти для вирішення цих проблем. Сучасні AI-моделі, зокрема великі мовні модел, здатні генерувати персоналізований контент, вести осмислений діалог у текстовому форматі, виправляти помилки та адаптуватися до рівня знань користувача [1]. Це створює унікальну можливість для розробки інтерактивних навчальних платформ, які імітують листування з носієм мови та надають вправи, що відповідають індивідуальним потребам.

Актуальність даної роботи полягає у необхідності створення гнучкого, доступного та ефективного інструменту для вивчення англійської мови, який би поєднував структуровані вправи з можливістю вільної текстової практики. Розробка веб-додатку з інтегрованим AI чат-ботом дозволяє вирішити проблему "«мовного бар'єру", надаючи користувачам безпечне середовище для тренування навичок письмової комунікації в будь-який час та з будь-якого місця. Такий підхід є особливо актуальним в умовах дистанційного навчання та зростаючої потреби у самоосвіті.

Метою курсового проєкту є проєктування та розробка інтерактивного веб-додатку для вивчення англійської мови, що надає користувачам

персоналізований досвід навчання через адаптивні вправи та взаємодію в чаті з AI-асистентом.

Для досягнення поставленої мети необхідно вирішити низку завдань. По-перше, провести аналіз предметної галузі освітніх веб-платформ та існуючих аналогів для виявлення їхніх переваг та недоліків. По-друге, сформулювати функціональні та нефункціональні вимоги до майбутнього програмного продукту. Важливою частиною роботи є проєктування архітектури клієнт-серверного застосунку, включаючи розробку структури бази даних для зберігання інформації про користувачів, навчальні матеріали та персональні словники. Основні етапи реалізації включають розробку серверної частини на платформі Node.js для забезпечення логіки авторизації та взаємодії з базою даних, а також розробку клієнтської частини з використанням бібліотеки React. Ключовим завданням є інтеграція генеративного штучного інтелекту через Google Gemini API для забезпечення функціонала AI-чат-бота. Завершальним етапом є тестування розробленого програмного продукту для перевірки його відповідності поставленим вимогам.

Галузь застосування результатів роботи є широкою та охоплює сферу онлайн-освіти. Розроблений веб-додаток може використовуватися як самостійний інструмент для самоосвіти, так і допоміжний засіб у межах формального навчання в школах, університетах чи на мовних курсах. Проєкт може бути корисним для студентів, школярів та дорослих, що прагнуть покращити свій рівень володіння англійською мовою.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Предметною галуззю даного курсового проєкту є ринок цифрових освітніх технологій, а саме сегмент програмних продуктів для самостійного вивчення іноземних мов. В умовах сучасної глобалізації володіння англійською мовою перетворилося з додаткової навички на ключову компетенцію, необхідну для успішної професійної діяльності, академічного росту, подорожей та міжкультурної комунікації. Це стимулює стабільно високий попит на ефективні, доступні та гнучкі інструменти для вивчення мов, що робить дану галузь однією з найбільш динамічних та конкурентних у сфері розробки програмного забезпечення.

Глобальний ринок онлайн-вивчення мов демонструє експоненційне зростання. Згідно з прогнозами аналітичних агенцій, таких як HolonIQ та Research and Markets, очікується, що обсяг ринку до 2027 року перевищить 20 мільярдів доларів США, із середньорічним темпом росту близько 18-20%. Ключовими драйверами цього росту є поширення смартфонів, доступ до високошвидкісного інтернету та прискорений перехід до дистанційних форматів навчання, спричинений глобальними подіями, зокрема пандемією COVID-19. Сучасні користувачі шукають не просто цифрові аналоги підручників, а повноцінні екосистеми, що пропонують персоналізований та інтерактивний досвід.

Сучасні платформи для вивчення мов можна класифікувати за основними методологіями, що лежать в їх основі. Перший підхід, гейміфікація, є найпопулярнішим для залучення масової аудиторії. Платформи, що його використовують, такі як Duolingo, Memrise, Drops, перетворюють процес навчання на гру. Вони використовують ігрові механіки, такі як нарахування балів, отримання досягнень, щоденні завдання, змагання з друзями та віртуальні нагороди. Основна мета гейміфікації – підвищити мотивацію та залученість користувача, перетворивши рутинне запам’ятовування слів на цікавий процес. Такі системи є надзвичайно ефективними для залучення нових користувачів та підтримки регулярності занять, особливо на початкових етапах вивчення мови.

Другий підхід базується на системах інтервального повторення. Ця методологія ґрунтується на науково доведеному ефекті психологічного інтервалу, який стверджує, що інформація краще засвоюється, якщо її повторювати через певні, зростаючі проміжки часу. Застосунки, що використовують метод інтервальних повторень, є потужними інструментами для довготривалого запам'ятовування великих обсягів лексики. Користувач створює або завантажує набори карток, а алгоритм сам визначає, коли яку картку потрібно повторити. Це високоефективний метод для розширення словникового запасу, але він не вирішує проблему активного використання слів у мовленні.

Третій підхід – комунікативний. Він реалізується через платформи, що з'єднують учнів з професійними репетиторами або носіями мови для практики розмовних навичок (italki, Preply, Cambly). Цей метод забезпечує найвищу якість навчання, оскільки дозволяє отримувати миттєвий зворотний зв'язок від живої людини, практикувати спонтанне мовлення та подолати мовний бар'єр. Недоліками є висока вартість та необхідність заздалегідь планувати заняття, що зменшує гнучкість навчання.

Четвертий підхід – це структуровані інтерактивні курси. Такі сервіси, як Babbel, Busuu, Lingualeo, пропонують навчальні програми, що нагадують традиційні мовні курси. Вони містять уроки з граматики, лексичні добірки, аудіо- та відеоматеріали, а також вправи на закріплення. Часто вони включають елементи комунікативної практики, наприклад, можливість надсилати свої письмові роботи на перевірку іншим членам спільноти.

Ключовим трендом, що об'єднує та вдосконалює ці підходи, є інтеграція технологій штучного інтелекту. AI використовується для персоналізації завдань, аналізу вимови та, що найважливіше, для реалізації чат-ботів. Використання великих мовних моделей, таких як Google Gemini, дозволяє створювати реалістичні діалоги [2], де AI виступає в ролі мовного партнера. Саме на перетині структурованих вправ та інтерактивної взаємодії з AI і знаходиться предметна область даного проєкту.

1.2 Виявлення та вирішення проблем

Незважаючи на велику кількість існуючих рішень, аналіз предметної галузі та відгуків користувачів дозволяє виявити низку системних проблем, які залишаються актуальними для значної частини учнів.

Основною проблемою є так званий розрив між знанням та вмінням. Більшість платформ успішно навчають користувачів лексики та граматичних правил, але не забезпечують достатньо практики для перетворення цих пасивних знань на активні навички. Користувач може мати словниковий запас у кілька тисяч слів, але не вміти побудувати просте речення в реальній розмові. Це відбувається через надмірний фокус на вправах з обмеженим вибором, де потрібно лише впізнати правильну відповідь, а не самостійно її згенерувати. Розроблюваний проєкт безпосередньо вирішує цю проблему, пропонуючи вправи на побудову речень та переклад, які вимагають від користувача активного застосування знань. Ключовим елементом є AI-чат-бот, що створює середовище для вільного текстового спілкування, де користувач змушений самостійно формулювати думки, що є найефективнішим способом тренування навичок.

Другою важливою проблемою є недостатня персоналізація та відсутність гнучкості. Багато застосунків, особливо гейміфіковані, пропонують жорсткий, лінійний шлях навчання, що не дозволяє користувачеві адаптувати програму під свої особисті цілі, наприклад, підготовку до подорожі або співбесіди. Рішенням у даному проєкті є надання користувачеві повного контролю над вибором теми та рівня складності. Це дозволяє AI генерувати релевантні завдання, роблячи навчання максимально цілеспрямованим.

Третьою проблемою є психологічний аспект – страх помилки. Багато учнів відчувають дискомфорт та невпевненість при текстовій комунікації з носіями мови, побоюючись зробити граматичну чи лексичну помилку. Цей мовний бар’єр значно сповільнює прогрес. AI-чат-бот виступає ідеальним тренажером, оскільки є неупередженим та терплячим партнером по листуванню, у діалозі з яким можна

безпечно експериментувати з мовою та отримувати конструктивний зворотний зв’язок у текстовому форматі.

Проблема втрати мотивації через монотонність є актуальною для багатьох платформ. Для її вирішення проєкт пропонує комбінацію кількох типів структурованих вправ та вільного чату. Можливість запросити у бота нову, унікальну вправу на обрану тему додає елемент непередбачуваності та підтримує інтерес до навчання.

1.3 Аналіз аналогів програмного забезпечення

Для розуміння ринкового середовища та визначення унікальності розроблюваного продукту було проведено детальний аналіз популярних аналогів, що представляють різні підходи до вивчення мов. До розгляду були включені Duolingo, Babbel, Memrise, Busuu та використання ChatGPT як самостійного інструменту.

Duolingo є абсолютним лідером ринку за кількістю завантажень та активних користувачів, що досягається завдяки потужній гейміфікації. Платформа перетворює навчання на гру з балами, лігами та щоденними завданнями для підтримки мотивації. Цей підхід є надзвичайно ефективним для залучення початківців та формування стабільної звички до навчання. Сильною стороною є також наявність повністю безкоштовної версії. Однак, з методичної точки зору, Duolingo часто критикують за поверхневе пояснення граматики та використання відірваних від реального життя речень, що ускладнює застосування знань на практиці. Можливості для вільного спілкування обмежені, а AI-функціонал доступний переважно у найдорожчій платній підписці.

На відміну від Duolingo, Babbel пропонує більш структурований та академічний підхід. Контент розробляється професійними лінгвістами, а уроки побудовані навколо реалістичних діалогів, що готує користувача до практичних ситуацій, таких як замовлення в ресторані або спілкування в аеропорту. Платформа надає глибокі та зрозумілі пояснення граматичних правил. Головними

недоліками є повністю платна модель розповсюдження за підпискою та менша гнучкість у виборі тем. Практика спілкування обмежена заздалегідь прописаними сценаріями, що не сприяє розвитку спонтанного мовлення.

Memrise поєднує в собі систему інтервального повторення для запам’ятовування слів та використання коротких відео з носіями мови. Основний фокус робиться на вивченні лексики в реальному контексті. Відеоролики допомагають почути, як слова звучать у повсякденному житті, що є значною перевагою. Однак Memrise менше уваги приділяє граматиці та побудові складних речень. Хоча платформа ефективна для розширення словникового запасу, вона не надає інструментів для вільної текстової практики чи глибокого розбору граматичних конструкцій.

Busuu є гібридною платформою, що поєднує структуровані уроки, схожі на Babbel, з унікальною соціальною функцією. Користувачі можуть надсилати свої письмові та усні вправи на перевірку носіям мови, які є членами спільноти, і отримувати від них виправлення та коментарі. Це створює елемент реальної комунікації та надає цінний зворотний зв’язок. Недоліком є те, що перевірка залежить від активності інших користувачів і може бути не миттєвою.

Використання великих мовних моделей, таких як ChatGPT, як інструменту для мовної практики, є потужним підходом. Його головна перевага – неперевершена гнучкість. AI може вести діалог на будь-яку тему, пояснювати граматичні правила, перекладати, перевіряти тексти та генерувати унікальні вправи. Це ідеальний інструмент для просунутих користувачів. Однак для початківців відсутність структури є значним недоліком. ChatGPT не веде користувача за визначеним планом, не відстежує прогрес та не пропонує інтегрованої системи вправ та словника, що вимагає від користувача високого рівня самоорганізації.

Аналіз аналогів показує, що на ринку існує незайнята ніша для продукту, який би поєднував найкращі риси існуючих рішень: структуру та методичність курсів, гнучкість та інтерактивність AI-чат-ботів, а також різноманітність вправ для підтримки мотивації. Розроблюваний веб-додаток займає саме цю нішу,

пропонуючи користувачеві "золоту середину". На відміну від лінійних курсів Duolingo чи Babbel, проєкт надає користувачеві свободу вибору теми для практики. Порівняно з соціальною моделлю Busuu, він гарантує миттєвий зворотний зв’язок від AI, доступний 24/7. А на відміну від використання ChatGPT як окремого інструменту, наш додаток інтегрує чат, структуровані вправи та персональний словник в єдину, зручну для користувача екосистему. Таким чином, ключовою перевагою проєкту є поєднання структурованого підходу до навчання з гнучкістю та інтерактивністю, що забезпечується сучасними технологіями штучного інтелекту.

2 ПОСТАНОВКА ЗАДАЧІ

Основна задача полягає в розробці інтерактивного веб-додатку для вивчення англійської мови, який надає користувачам можливість навчатися за індивідуальною траєкторією. Система має оперувати ключовими об’єктами предметної області, такими як користувачі, їхні рівні знань, навчальні вправи та персональні словники.

Перелік задач для реалізації програмної системи включає:

– реалізацію серверної частини (backend) для управління логікою додатку, включаючи API для взаємодії з клієнтською частиною;

– розробку клієнтської частини (frontend) з інтуїтивно зрозумілим інтерфейсом для всіх функціональних модулів;

– реалізацію трьох типів інтерактивних вправ: вибір правильної відповіді, побудова речення та переклад слів;

– інтеграцію зовнішнього сервісу генеративного штучного інтелекту для забезпечення функціонала AI-чат-бота;

– проведення тестування основних сценаріїв взаємодії користувача з системою.

Інформація для бази даних повинна включати:

– структуру питань для вправ: текст питання, тип вправи, рівень складності, варіанти відповідей у форматі JSON (для відповідного типу), правильна відповідь, слово для вивчення.

Обов’язкові функції програмної системи:

– адаптивна вибірка вправ з бази даних відповідно до рівня знань, вказаного користувачем у профілі;

– надання миттєвого візуального зворотного зв’язку (підсвічування правильних/неправильних відповідей);

– ведення рахунку (балів) користувача в реальному часі під час проходження тесту;

– можливість вибору теми та рівня складності для практики в чаті з AI;

– генерація унікальних завдань AI-асистентом за запитом користувача на основі обраної теми;

– збереження слів до персонального словника користувача із перевіркою на дублікати.

Формулювання запитів до інформаційної системи:

– вибірка 10 випадкових питань певного типу та рівня складності;

– пошук користувача в базі даних за його ідентифікатором для отримання рівня;

– додавання нового слова до словника конкретного користувача;

– перевірка наявності слова у словнику користувача перед додаванням.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

У рамках UML-проєктування першою була створена діаграма прецедентів (рис. 3.1), що дозволяє описати функціональні вимоги до веб-додатку з точки зору користувача. Вона відображає ключові сценарії взаємодії між зовнішніми акторами та системою, а також окреслює основні функції, які реалізуються у проєкті.

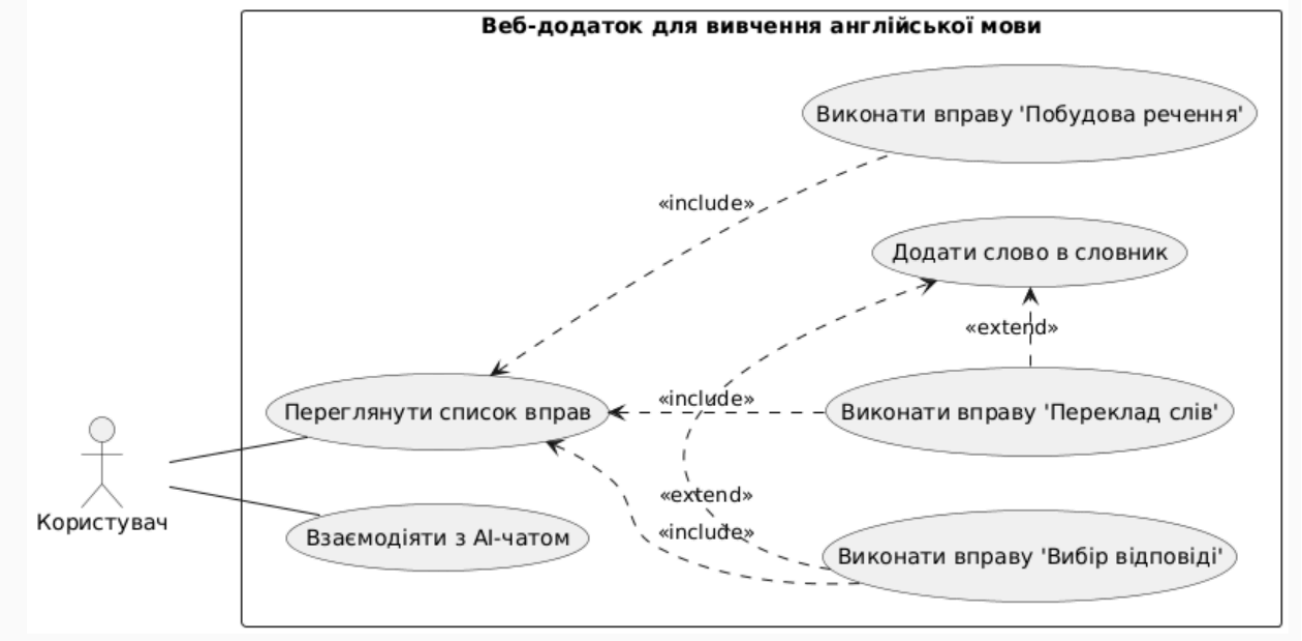


Рисунок 3.1 – Діаграма прецедентів веб-додатку

На діаграмі визначено одного головного актора – "Користувач". Він ініціює два основні сценарії взаємодії: "Переглянути список вправ", що є точкою входу до структурованого навчання, та "Взаємодіяти з AI-чатом" для вільної практики.

Прецедент "Переглянути список вправ" є базовим для всіх трьох типів вправ. Зв’язок <<include>>, що йде від прецедентів "Виконати вправу 'Побудова речення'", "Виконати вправу 'Переклад слів'" та "Виконати вправу 'Вибір відповіді'", вказує на те, що виконання будь-якої вправи є неможливим без попереднього перегляду їх загального списку.

Прецедент "Додати слово в словник" пов’язаний з прецедентами "Виконати вправу 'Переклад слів'" та "Виконати вправу 'Вибір відповіді'" за допомогою зв'язку <<extend>>. Це означає, що функція додавання слова є необов’язковим розширенням, яке стає доступним користувачеві під час виконання саме цих вправ.

Для деталізації динамічної поведінки інтерактивних навчальних модулів було розроблено узагальнену діаграму станів (рис. 3.2). Вона моделює життєвий цикл одного завдання з двох типів вправ (вибір відповіді, переклад слів) від моменту його відображення до переходу до наступного, ілюструючи реакцію системи на дії користувача.

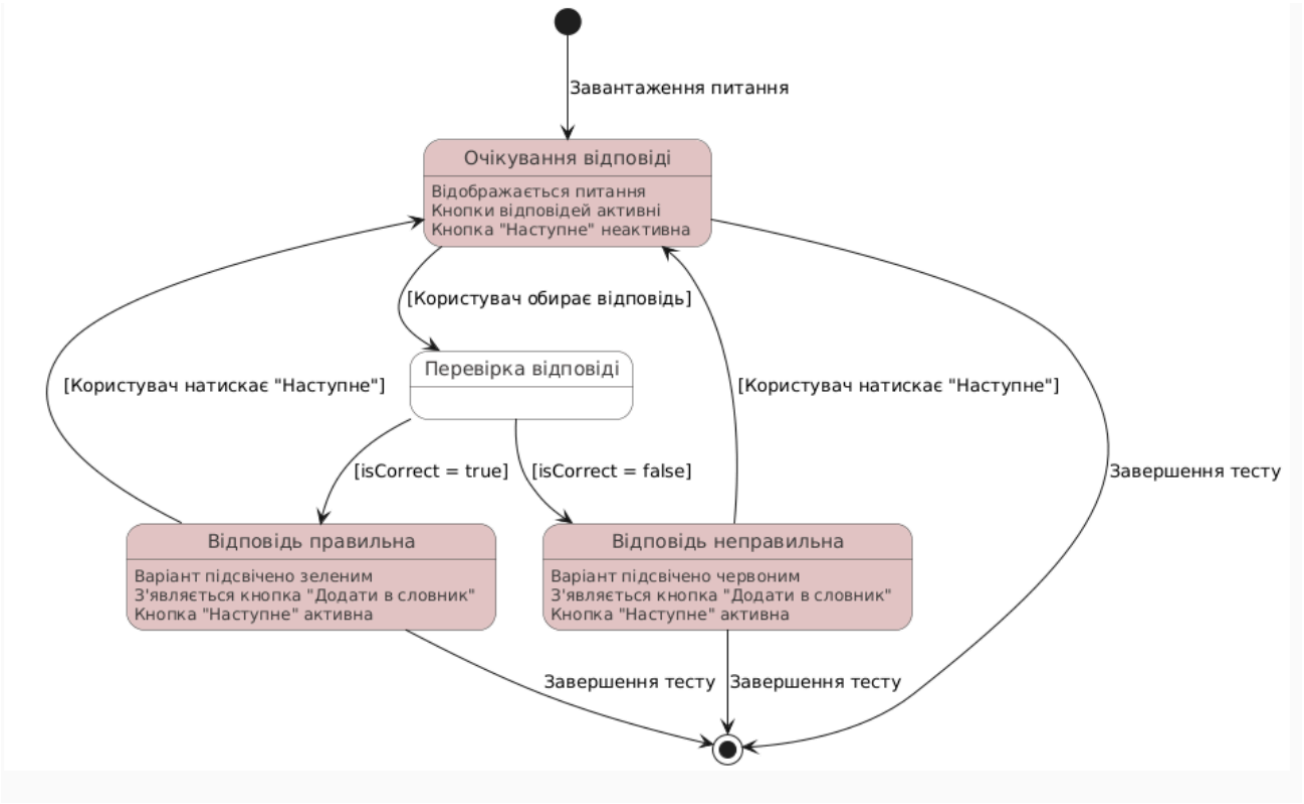


Рисунок 3.2 – Діаграма станів для одного питання вправи

Життєвий цикл починається із завантаження питання, після чого система переходить у стан "Очікування відповіді". У цьому стані користувачеві відображається текст питання та активні кнопки варіантів відповіді, тоді як кнопка "Наступне" є неактивною.

Коли користувач обирає один із варіантів, відбувається перехід у проміжний стан "Перевірка відповіді". Залежно від результату перевірки, система переходить в один із двох кінцевих станів для поточного питання: "Відповідь правильна" або "Відповідь неправильна". В обох цих станах обраний варіант візуально підсвічується, з’являється кнопка "Додати в словник", а кнопка "Наступне" активується, дозволяючи користувачеві продовжити. Після натискання кнопки "Наступне", система або повертається у стан "Очікування відповіді" з новими даними, або якщо це було останнє питання, завершує тест.

3.2 Проєктування архітектури ПЗ

Проєктування архітектури є важливим етапом створення програмного забезпечення, що визначає його структуру, компоненти та принципи функціонування. Для реалізації поставлених задач, а саме розробки інтерактивного веб-додатку для вивчення англійської мови, було проведено архітектурний огляд та обрано оптимальну модель. Вибір архітектурного стилю є стратегічним рішенням, що впливає на подальшу розробку, масштабованість та підтримуваність системи.

Для даного проєкту було обрано класичну клієнт-серверну архітектуру [7], оскільки вона найкраще відповідає природі веб-додатків. Цей архітектурний стиль дозволяє чітко розмежувати відповідальність між клієнтською логікою, що виконується у браузері користувача і відповідає за користувацький інтерфейс, та серверною логікою, яка керує даними, автентифікацією та взаємодією зі сторонніми сервісами. Це забезпечує високу гнучкість, масштабованість та можливість незалежної розробки й оновлення кожної частини системи.

Система логічно поділена на три рівні, що відповідає патерну багаторівневої архітектури [3]:

– презентаційний рівень (Frontend) відповідає за відображення інформації та взаємодію з користувачем. На цьому рівні реалізовано інтерфейс користувача, а також здійснюється обробка введених даних;

– рівень бізнес-логіки (Backend) забезпечує обробку запитів, реалізацію основної функціональності додатку, а також взаємодію з базою даних і зовнішніми сервісами;

– рівень даних (Data Layer) відповідає за зберігання, обробку та доступ до структурованих даних, включаючи виконання CRUD-операцій і підтримку цілісності інформації.

Такий підхід базується на принципі розділення відповідальності, що є одним з ключових принципів проєктування ПЗ [8]. Кожен рівень має свою чітко визначену роль і не втручається в роботу інших, що значно спрощує розробку, тестування та подальшу підтримку коду. Для візуалізації цієї структури було спроєктовано UML діаграму компонентів (рис. 3.3).

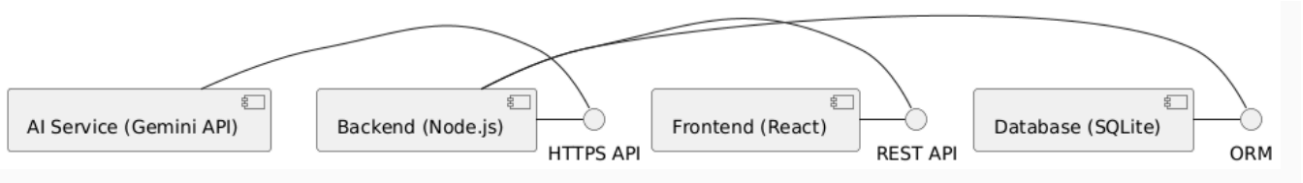


Рисунок 3.3 – Діаграма компонентів системи

Клієнтський додаток реалізований як односторінковий додаток з використанням бібліотеки React. Вибір React зумовлений його компонентним підходом, що дозволяє створювати складні інтерфейси з пере-використовуваних блоків [4]. Управління станом реалізовано за допомогою вбудованих хуків React (useState, useEffect), що дозволяє ефективно керувати даними на клієнті без залучення складних сторонніх бібліотек для управління станом, що є оптимальним для проєкту даного масштабу. Навігація між сторінками (WordsPage, MultipleChoiceTest тощо) реалізована за допомогою бібліотеки react-router-dom, що забезпечує швидкі переходи без перезавантаження сторінки.

Сервер додатку розроблений на платформі Node.js з використанням фреймворку Express. Node.js був обраний завдяки своїй асинхронній, неблокуючій моделі вводу-виводу, що ідеально підходить для обробки великої кількості одночасних HTTP-запитів. Express [5] надає мінімалістичний, але потужний набір

інструментів для створення REST API. Структура серверу є модульною: логіка для кожного типу сутностей (вправи, словник) винесена в окремі файли маршрутизаторів (exercises.js, dictionary.js), що відповідає принципу єдиної відповідальності.

Шар доступу до даних реалізований за допомогою ORM Sequelize [6]. Цей компонент виступає в ролі "перекладача" між об'єктно-орієнтованим кодом JavaScript та реляційною базою даних SQLite. Використання ORM дозволяє працювати з даними як з об’єктами, що робить код більш читабельним та менш залежним від конкретної СУБД. Моделі Sequelize (Question, UserDictionary) описують структуру таблиць та зв’язки між ними, забезпечуючи валідацію даних на рівні моделі.

Механізми комунікації між компонентами чітко стандартизовані. Взаємодія між клієнтом та сервером відбувається виключно через REST API по протоколу HTTPS. Асинхронна природа fetch-запитів на клієнті дозволяє інтерфейсу залишатися відзивчивим під час очікування відповіді від сервера. Для автентифікації використовується стандарт JWT.

Для розуміння фізичного розміщення компонентів було спроєктовано UML діаграму розгортання (рис. 3.4).

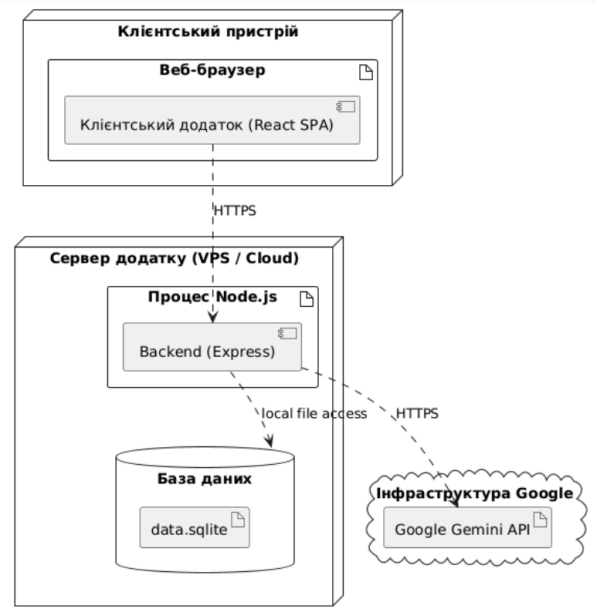


Рисунок 3.4 – Діаграма розгортання системи

Діаграма ілюструє, що клієнтський додаток виконується у веб-браузері на пристрої користувача. Сервер додатку, що включає процес Node.js та файл бази даних SQLite, розгортається на єдиному серверному вузлі. Таке рішення є оптимальним для початкового етапу проєкту, оскільки спрощує налаштування та розгортання. Google Gemini API функціонує на зовнішній хмарній інфраструктурі Google, і взаємодія з ним відбувається через зашифроване HTTPS-з'єднання.

При проєктуванні були враховані і нефункціональні вимоги. Для забезпечення продуктивності відповіді сервера на запити не повинні перевищувати 500 мс. Для підвищення надійності система коректно обробляє помилки, наприклад, недоступність зовнішнього AI-сервісу, та інформує про них користувача.

Також було проведено оцінку ризиків. Основний ризик це залежність від зовнішнього сервісу Google Gemini API. У випадку його недоступності або зміни умов використання, функціонал чату перестане працювати. Для зменшення цього ризику логіка взаємодії з AI інкапсульована в окремому сервісному модулі, що в майбутньому дозволить відносно легко замінити його на іншу мовну модель.

У результаті спроєктовано архітектуру інтерактивного веб-додатку для вивчення англійської мови, засновану на клієнт-серверній моделі з трирівневою структурою: презентаційна логіка реалізована на React, бізнес-логіка – Node.js, а зберігання даних у SQLite. Компоненти системи взаємодіють через REST API у форматі JSON з використанням JWT-автентифікації [10], а генерація контенту здійснюється через інтеграцію з Google Gemini API. Побудовані UML-діаграми узгоджуються з функціональними вимогами та підтверджують правильність обраної архітектури. Наступними етапами є реалізація клієнтської та серверної частин, тестування, а також розгортання додатку.

3.3 Проєктування структури зберігання даних

Для розроблюваного веб-додатку було обрано реляційну модель даних, оскільки вона забезпечує чітку структуру, цілісність даних та гнучкі можливості

для виконання запитів. В якості системи управління базами даних було обрано SQLite завдяки її простоті, відсутності необхідності в окремому серверному процесі та легкій інтеграції з Node.js, що є оптимальним для проєктів такого масштабу.

На основі аналізу предметної області та функціональних вимог було визначено три ключові сутності, які необхідно зберігати в базі даних:

Таблиця 3.1 – Структура таблиці «Users»

№	Найменування	Тип даних	Призначення
1	id	INTEGER	Унікальний ідентифікатор користувача
2	first_name	VARCHAR(255)	Ім’я користувача
3	second_name	VARCHAR(255)	Прізвище користувача
4	email	VARCHAR(255)	Електронна пошта
5	phone	VARCHAR(255)	Контактний номер
6	password	VARCHAR(255)	Хешований пароль
7	level	TEXT	Рівень володіння мовою
8	created_at	DATETIME	Час створення акаунту

Таблиця 3.2 – Структура таблиці «questions»

№	Найменування	Тип даних	Призначення
1	id	INTEGER	Унікальний ідентифікатор питання
2	exercise_type	VARCHAR(255)	Тип вправи (наприклад, multiple_choice)
3	question_text	TEXT	Текст питання
4	options	TEXT	Варіанти відповідей (для multiple_choice)
5	correct_answer	VARCHAR(255)	Правильна відповідь

Кінець таблиці 3.2

6	word_to_learn	VARCHAR(255)	Ключове слово або фраза для додавання до словника
7	translation	TEXT	Переклад слова на українську
8	difficulty_level	VARCHAR(255)	Рівень складності

Таблиця 3.3 – Структура таблиці «user_dictionary»

№	Найменування	Тип даних	Призначення
1	id	INTEGER	Унікальний ідентифікатор запису у словнику
2	user_id	INTEGER	Зовнішній ключ на користувача
3	word	VARCHAR(255)	Слово англійською
4	translation	VARCHAR(255)	Переклад українською
5	added_at	DATETIME	Дата й час додавання слова у словник

Для візуалізації спроєктованої структури наведено схему бази даних (рис. 3.5).

Основний зв’язок у базі даних існує між таблицями «Users» та «user_dictionary». Це зв’язок типу "один-до-багатьох". Один користувач може мати багато записів у своєму словнику, але кожен запис у словнику належить лише одному конкретному користувачеві. Цей зв’язок реалізується за допомогою зовнішнього ключа «user_id» в таблиці «user_dictionary», який посилається на первинний ключ id таблиці «Users». Таблиця questions слугує загальним банком завдань для всієї системи і не має жорстких фізичних зв’язків з іншими таблицями. Однак між нею та таблицею «user_dictionary» існує логічна залежність: дані з полів «word_to_learn» та «translation» таблиці «questions»

використовуються для створення нових записів у персональному словнику користувача.

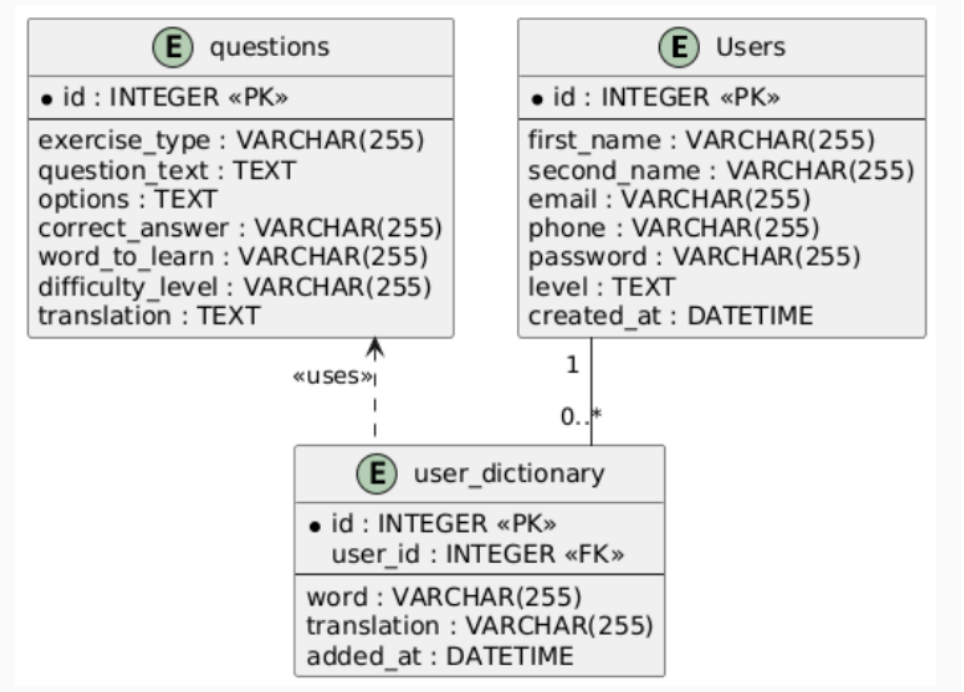


Рисунок 3.5 – Схема реляційної бази даних

3.4 Приклади використаних алгоритмів та методів

Основний алгоритм, що лежить в основі додатку це алгоритм взаємодії з AI чат-ботом. Він описує повний цикл обробки запиту користувача, від відправки повідомлення до отримання та відображення відповіді від штучного інтелекту. Цей процес найкраще ілюструє діаграма активностей (рис. 3.6), яка показує потік керування між клієнтом, сервером та зовнішнім AI-сервісом.

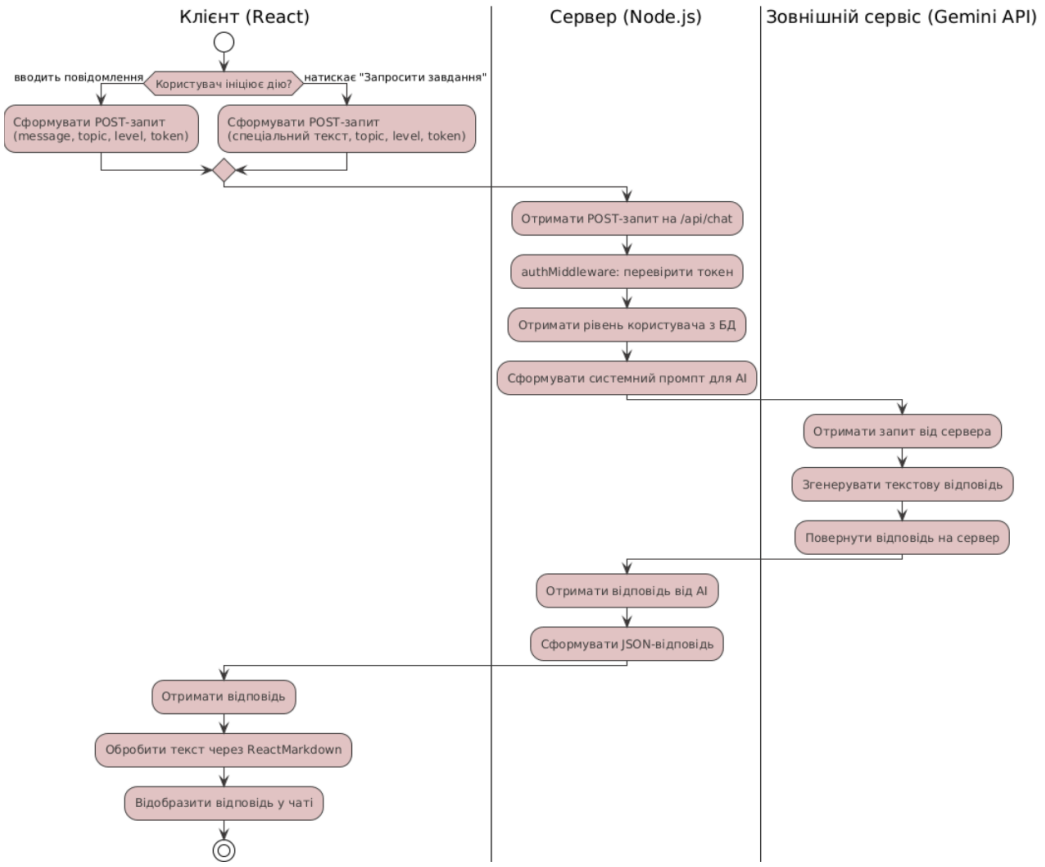


Рисунок 3.6 – Діаграма активностей для взаємодії з AI-чатом

Процес починається на клієнті, далі на сервері запит обробляється відповідним роутом. Спочатку витягується інформація про користувача для визначення його рівня знань. На основі повідомлення, теми та рівня, динамічно генерується комплексний системний запит для Google Gemini API. Сервер відправляє цей запит до зовнішнього AI-сервісу. Після отримання згенерованої текстової відповіді від AI, сервер формує JSON-відповідь і відправляє її назад на клієнт.

На клієнті отримана відповідь додається до історії чату, що викликає оновлення інтерфейсу. Для коректного відображення можливого форматування відповідь пропускається через бібліотеку react-markdown.

Важливою частиною цього алгоритму є можливість запиту на генерацію вправи. Коли користувач натискає кнопку "Запросити завдання", на сервер відправляється спеціально сформований текстовий запит. Серверний запит налаштований таким чином, щоб розпізнавати цей запит і змушувати AI

генерувати не діалогову відповідь, а унікальне мовне завдання, що відповідає обраній темі та рівню, таким чином інтегруючи навчальний процес безпосередньо в діалог.

Іншим важливим алгоритмом, що є центральним для всіх трьох типів інтерактивних вправ, є алгоритм обробки відповіді користувача. Його мета – визначення правильності відповіді, оновлення стану інтерфейсу та нарахування балів. Процес, що активується дією користувача, представлений на блок-схемі (рис. 3.7). Спочатку з поточного стану компонента отримуються дані про відповідь користувача та правильний варіант. Для забезпечення коректного порівняння обидва рядки нормалізуються, приводяться до нижнього регістру та очищуються від зайвих пробілів. Якщо відповідь правильна, система оновлює стан, нараховуючи бали, та застосовує до відповідного елемента інтерфейсу стиль, що вказує на успіх. В іншому випадку застосовується стиль для позначення помилки. Після перевірки система блокує можливість повторної відповіді та активує кнопку переходу до наступного завдання, а також надає користувачеві можливість додати вивчене слово до свого персонального словника.

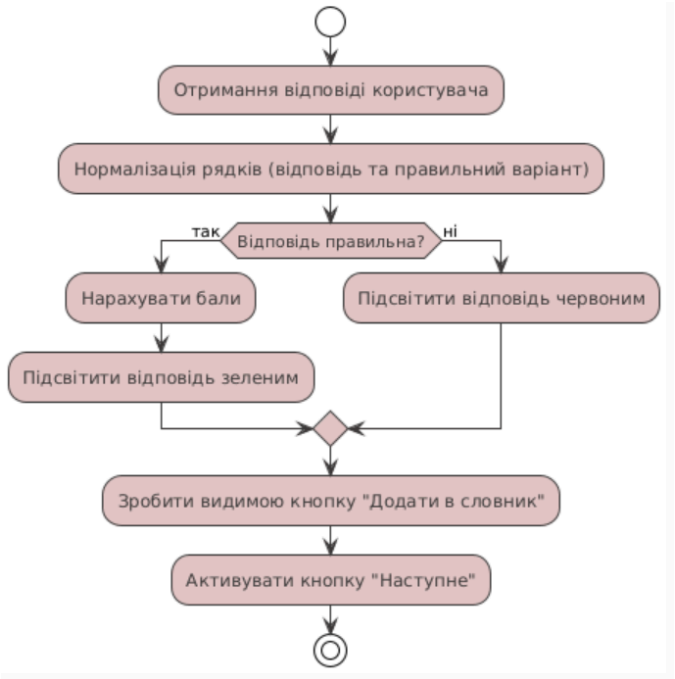


Рисунок 3.7 – Діаграма активностей для завантаження завдань

3.5 Проєктування UI/UX

Проєктування інтерфейсу користувача та організація користувацького досвіду становлять важливу складову створення сучасного веб-додатку. На цьому етапі особливу увагу було приділено розробці зрозумілого, естетично привабливого та функціонального середовища, яке сприяє залученню користувача до навчального процесу.

За основу концепції дизайну було взято мінімалістичний стиль. Для створення спокійної та невимушеної атмосфери, що сприяє навчанню, було обрано палітру з м'яких, пастельних кольорів. Основний рожевий фон створює відчуття комфорту, тоді як акцентний синій використовується для інтерактивних елементів, таких як кнопки вправ та повідомлення від AI, асоціюючись зі спокоєм та знаннями. Такий вибір кольорів дозволяє візуально розділити інформаційні блоки та інтерактивні елементи, ефективно направляючи увагу користувача.

Дизайн ключових сторінок було спроєктовано з урахуванням їх функціонального призначення. Сторінка вибору вправ (рис. 3.8) виступає в ролі навігаційного хабу і реалізована за картковим принципом. Кожна картка, що представляє окремий тип вправи, містить чітку іконку та назву, що дозволяє користувачеві швидко ідентифікувати потрібну активність.

Для забезпечення консистентності користувацького досвіду, сторінки виконання вправ (рис. 3.9-3.11) спроєктовані за єдиним шаблоном. Ключовими елементами на цих сторінках є заголовок з іконкою, що ідентифікує тип завдання, а також прогрес-бар та лічильник балів, які наочно демонструють прогрес користувача і слугують важливим елементом мотивації. Центральний блок завдання є фокусною точкою екрану, а кнопки дій, такі як "Перевірити" та "Наступне", розташовані внизу для зручності взаємодії.

Рисунок 3.8 – Інтерфейс сторінки вибору вправ

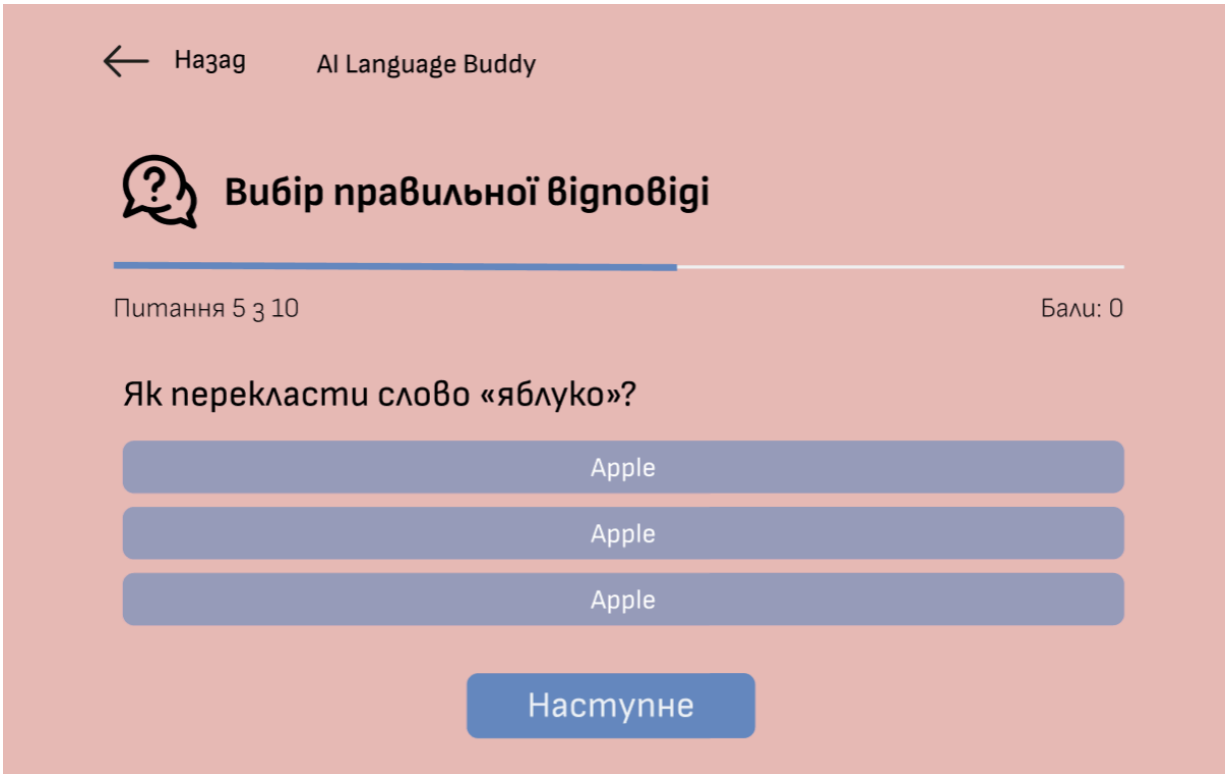


Рисунок 3.9 – Інтерфейс вправи з вибором правильного варіанту

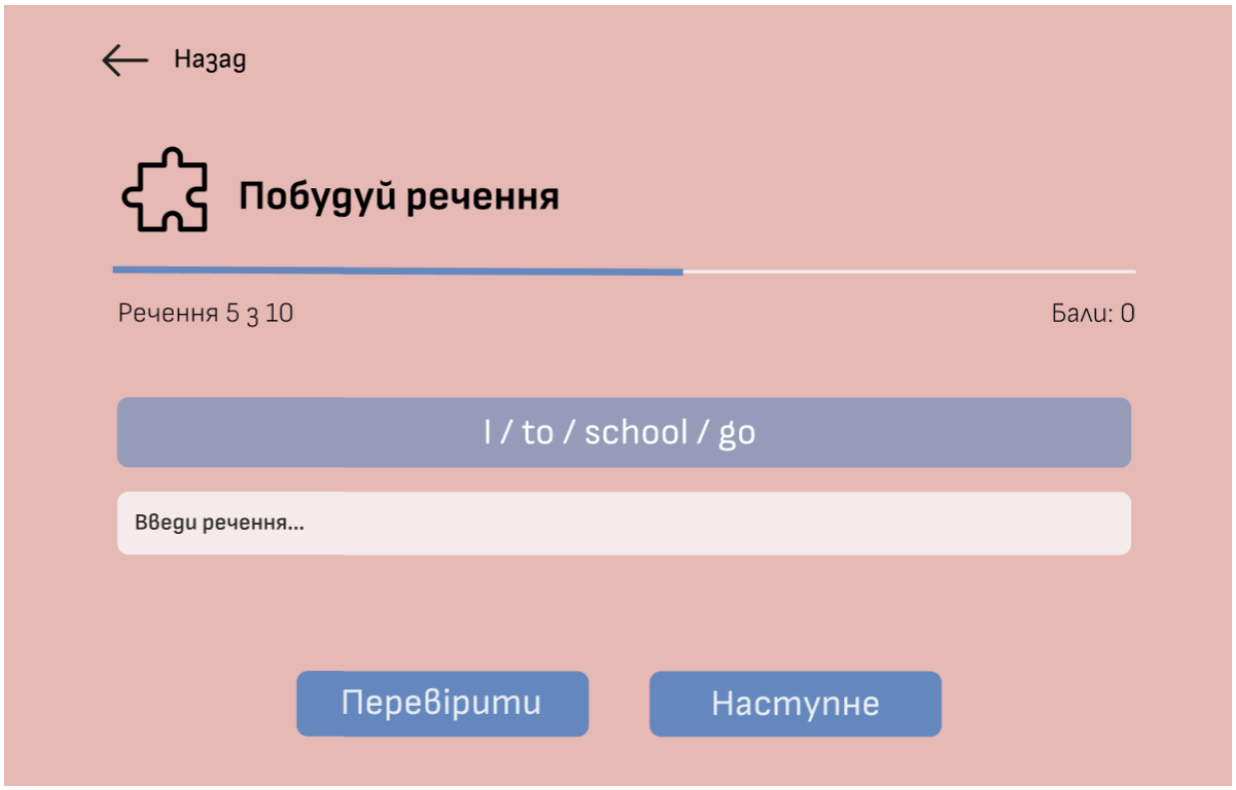


Рисунок 3.10 – Інтерфейс вправи з побудови речення

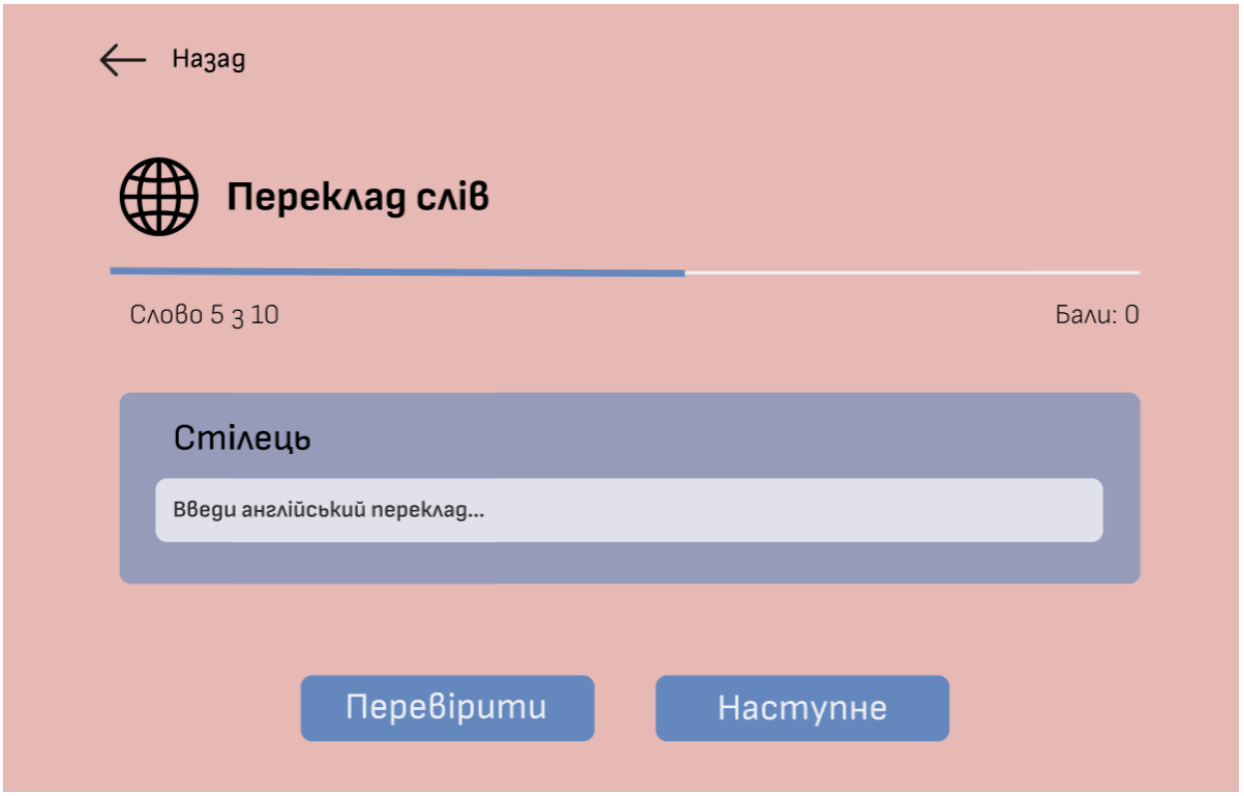


Рисунок 3.11 – Інтерфейс вправи з перекладу слів

Найбільш комплексним з точки зору дизайну є інтерфейс сторінки "AI Language Buddy" (рис. 3.12). Він логічно поділений на дві зони. Лівий сайдбар надає користувачеві інструменти для керування контекстом діалогу – вибір теми та рівня складності. Це дозволяє зробити взаємодію з AI більш цілеспрямованою. Права панель чату реалізована за класичною схемою месенджерів, де повідомлення користувача та бота візуально відрізняються за кольором та вирівнюванням для легкого читання. Додаткові підказки-запити під вікном чату допомагають користувачеві почати розмову.

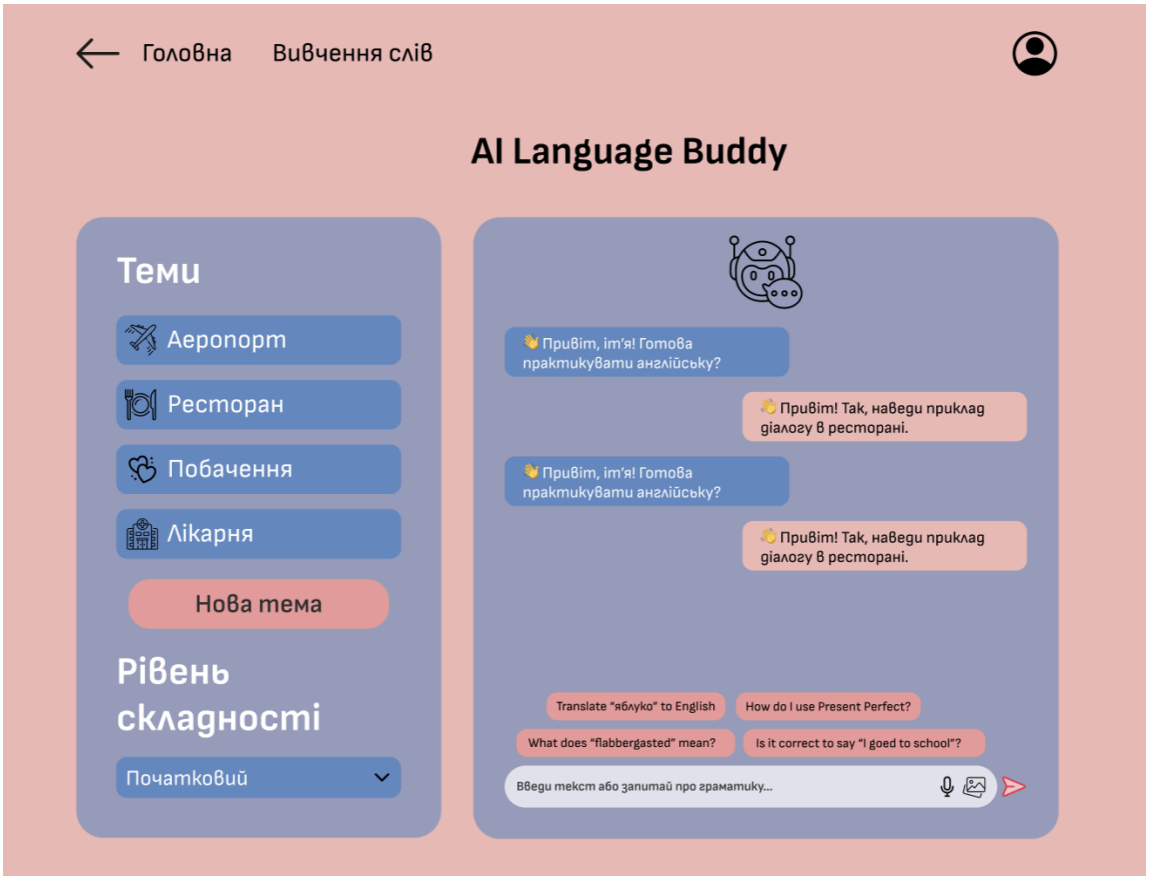


Рисунок 3.12 – Інтерфейс AI чат-бота

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

Для розробки програмного продукту було обрано набір технологій на основі JavaScript, що дозволило реалізувати повноцінний веб-додаток із інтерактивним інтерфейсом та потужною серверною логікою. Клієнтська частина створена з використанням бібліотеки React, компонентний підхід якої забезпечив розбиття інтерфейсу на невеликі, незалежні та багаторазово використовувані елементи. Це значно спростило процес розробки та подальше обслуговування коду. Для управління станом використовувались вбудовані React-хуки, зокрема useState та useEffect, а навігацію між сторінками реалізовано за допомогою бібліотеки react-router-dom.

Серверна частина базується на платформі Node.js у поєднанні з фреймворком Express, що забезпечує ефективну обробку асинхронних операцій, критично важливу умову для взаємодії із зовнішніми API та базою даних. Для зберігання інформації було використано легку у налаштуванні файлову СУБД SQLite. Робота з даними реалізована через ORM-бібліотеку Sequelize, яка забезпечує об’єктно-орієнтований підхід до доступу до бази даних та додатковий рівень абстракції.

Для впровадження функціоналу чат-бота система інтегрована з Google Gemini API (модель gemini-1.5-flash), що надає розширені можливості для генерації природного тексту та організації діалогу в межах навчального процесу.

Ключовим рішенням на фронтенді стало створення модульних та функціональних компонентів. Розглянемо реалізацію сторінки "Вибір правильної відповіді" (MultipleChoiceTest.js) як приклад (рис. 4.1).

Завантаження завдань з сервера відбувається асинхронно за допомогою хука useEffect при першому рендерингу компонента. Це дозволяє відображати індикатор завантаження, поки дані отримуються з сервера. Наведемо фрагмент коду, що відповідає за цей процес:

```
useEffect(() => {
    const fetchQuestions = async () => {
```

```
            const token = localStorage.getItem('token');
            if (!token) {
                setError("Доступ заборонено. Будь ласка, увійдіть в
систему.");
                setIsLoading(false);
                return;
            }
            try {
                const response = await
fetch('http://localhost:3001/api/exercises/multiple-choice', {
                    headers: { 'Authorization': `Bearer ${token}` }
                });
                if (!response.ok) throw new Error('Не вдалося завантажити
питання.');
                const data = await response.json();
                if (data.questions && data.questions.length > 0) {
                    setQuestions(data.questions);
                } else {
                    throw new Error('Для вашого рівня ще немає питань
цього типу.');
                }
            } catch (err) {
                setError(err.message);
            } finally {
                setIsLoading(false);
            }
        };
        fetchQuestions();
    }, []);
```

Після вибору відповіді користувачем викликається функція handleAnswerSelect, яка порівнює відповідь з правильною, оновлює стан (isCorrect, score) та змінює CSS-класи кнопок для візуальної індикації результату.

Рисунок 4.1 – Інтерфейс вправи "Вибір правильної відповіді"

На бэкенді ключовим рішенням було створення окремих маршрутизаторів для різних сутностей (exercises.js, dictionary.js, chat.js), що дозволило структурувати код. Розглянемо фрагмент роута, що відповідає за надання завдань для вправи "Побудова речення":

```
router.get('/sentence-builder', authenticate, async (req, res) => {
    try {
        const user = await User.findByPk(req.user.id);
        if (!user) return res.status(404).json({ error: "Користувач
не найден" });

        const userLevel = user.level.toLowerCase();

        const tasks = await Question.findAll({
            where: {
                exercise_type: 'sentence_builder',
                difficulty_level:
Sequelize.where(Sequelize.fn('lower', Sequelize.col('difficulty_level')),
userLevel)
            },
            order: [ Sequelize.fn('RANDOM') ],
            limit: 10
        });

        res.json({ tasks });
    } catch (err) {
        console.error("Error fetching sentence-builder tasks:", err);
        res.status(500).json({ error: "Помилка сервера" });
    }
});
```

Даний код демонструє використання проміжного обробника authenticate для перевірки токена, отримання рівня користувача з бази даних та виконання запиту за допомогою Sequelize з умовою, не чутливою до регістру (Sequelize.fn('lower', ...)), що підвищує надійність системи.

Вправа "Побудова речення" вимагає від користувача більш активних дій. Інтерфейс (рис. 4.2) надає набір слів у довільному порядку та текстове поле для вводу.

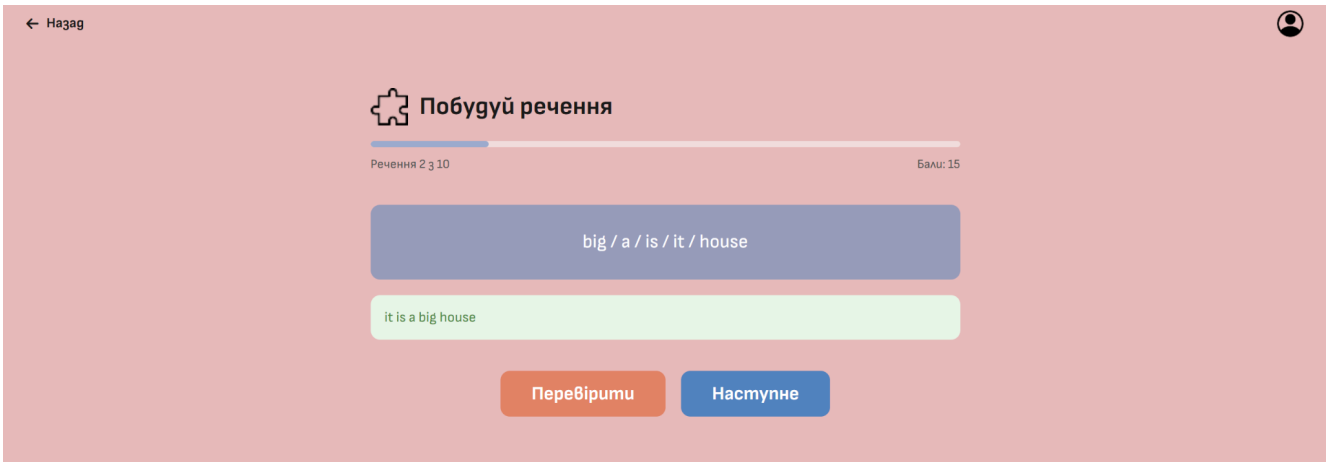


Рисунок 4.2 – Інтерфейс вправи "Побудова речення"

Задача користувача: скласти з цих слів граматично правильне речення. Механізм перевірки, реалізований у функції handleCheckAnswer, є більш комплексним. Він нормалізує рядок, введений користувачем, та правильну відповідь, приводячи їх до нижнього регістру та видаляючи знаки пунктуації. Щоб уникнути помилок через незначні розбіжності, наприклад, відсутність крапки в кінці речення. Розглянемо код функції handleCheckAnswer:

```
const handleCheckAnswer = () => {
    const currentTask = tasks[currentIndex];
    if (!currentTask) return;

    // Нормалізуємо рядки для надійного порівняння
    const userInput =
inputValue.trim().toLowerCase().replace(/[.,/#!$%^&*;:{}=\-_`~()?]/g,
"");
    const correctAnswer =
currentTask.correct_answer.trim().toLowerCase().replace(/[.,/#!$%^&*;
:{}=\-_`~()?]/g, "");

    if (userInput === correctAnswer) {
        setIsCorrect(true);
        setScore(prev => prev + 15);
    } else {
        setIsCorrect(false);
    }
    setIsChecked(true);
};
```

Вправа "Переклад слів" спрямована на перевірку та закріплення словникового запасу (рис. 4.3). Користувачеві демонструється слово українською мовою, а він має ввести його англійський переклад. Ключовою особливістю є

функція "Додати до словника". Після надання відповіді з'являється кнопка, що дозволяє відправити асинхронний POST-запит на сервер для збереження слова у персональному словнику користувача:

```
    const handleAddToDictionary = async (word, translation) => {
        const token = localStorage.getItem('token');
        try {
            const response = await
fetch('http://localhost:3001/api/dictionary/add', {
                method: 'POST',
                headers: { 'Content-Type': 'application/json',
'Authorization': `Bearer ${token}` },
                body: JSON.stringify({ word: word, translation:
translation })
            });
            const result = await response.json();
            if (!response.ok) {
                throw new Error(result.message || 'Не вдалося додати
слово');
            }
            alert(`Слово "${word}" успішно додано!`);
        } catch (err) {
            alert(err.message);
        }
    };
```

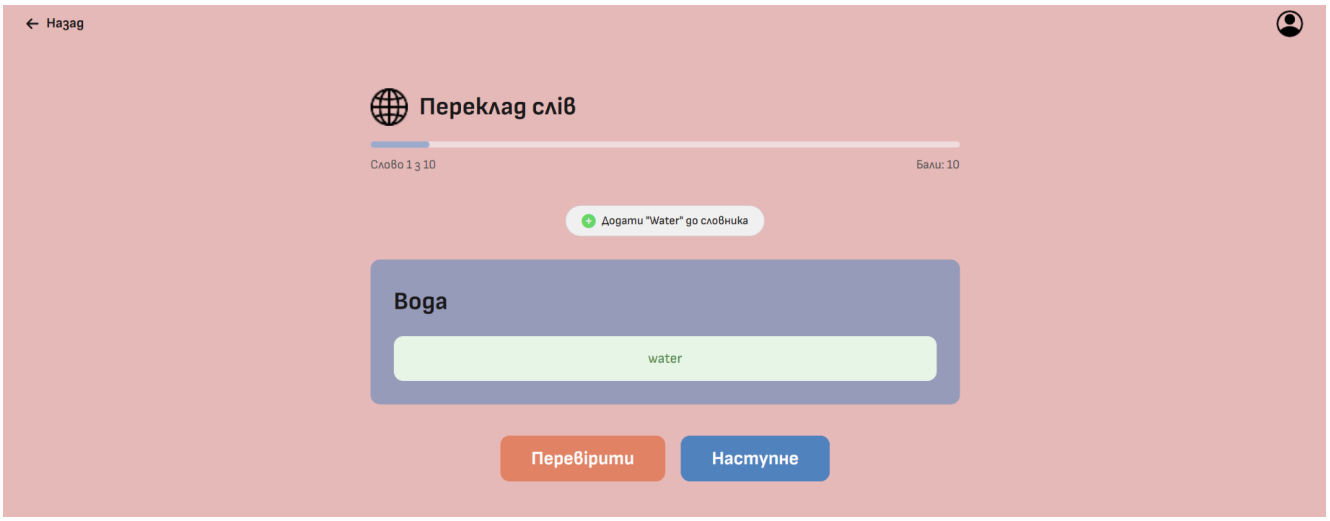


Рисунок 4.3 – Інтерфейс вправи "Побудова речення"

Найбільш прогресивним рішенням є інтеграція з Google Gemini API. Вся логіка взаємодії з AI інкапсульована в роуті /api/chat. Цей маршрут приймає від клієнта текст повідомлення, обрану тему та рівень, після чого динамічно формує детальний системний запит. Такий підхід дозволяє гнучко керувати поведінкою

AI-моделі, змушуючи її відповідати виключно англійською мовою та генерувати вправи у потрібному форматі.

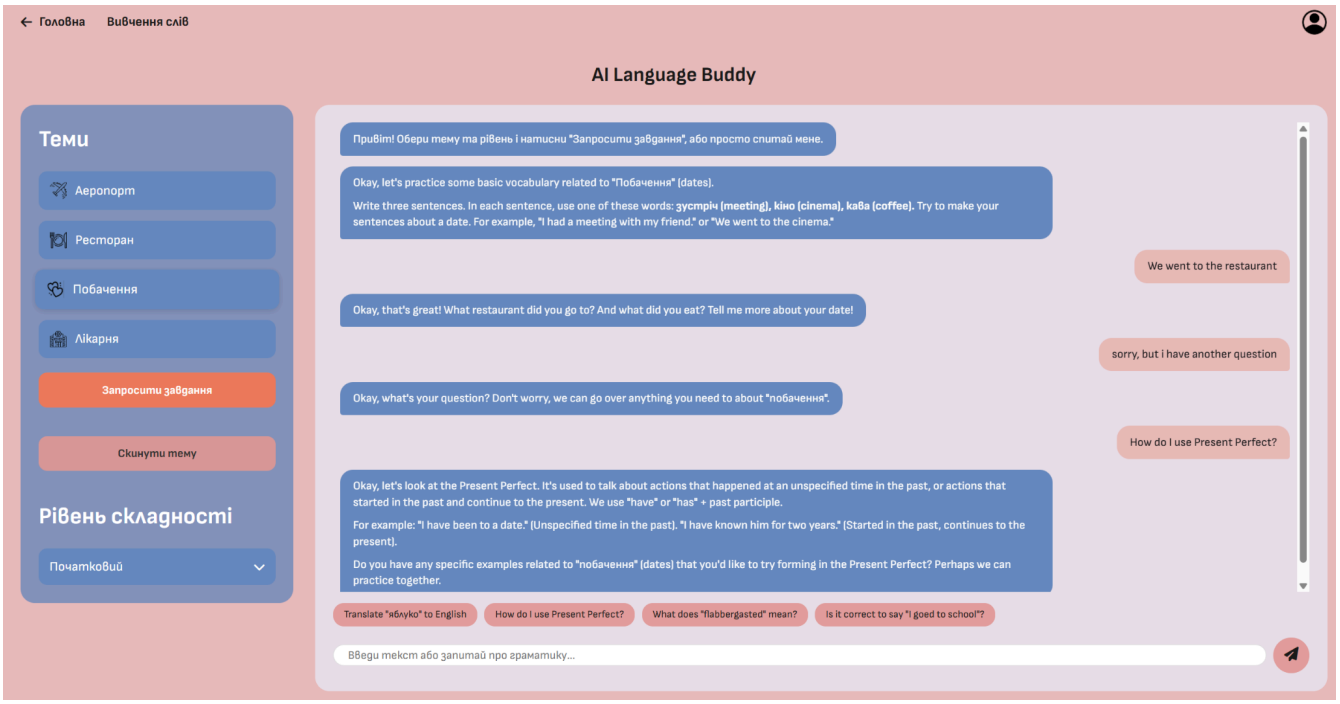


Рисунок 4.4 – Результат взаємодії з AI-асистентом

На рисунку 4.4 приклад відповіді AI на запит користувача. Для коректного відображення форматування, отриманого від AI (заголовки, списки, жирний текст), на клієнтській стороні використовується бібліотека react-markdown, яка перетворює Markdown-розмітку на HTML-теги. Це дозволяє отримувати від AI структурований та візуально привабливий контент.

5 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

5.1 Порівняння результатів із початковими вимогами

Першим кроком аналізу є порівняння фінального продукту з вимогами, сформульованими на етапі постановки задачі. Основною метою проєкту було створення інтерактивного середовища, що поєднує структуровані вправи та вільну практику з AI-асистентом. Цю мету було повністю досягнуто. Було реалізовано п'ять ключових сторінок, що складають ядро додатку: головна сторінка вибору вправ, три сторінки з різними типами вправ ("Вибір відповіді", "Побудова речення", "Переклад слів") та сторінка AI чат-бота.

Функціональні вимоги були реалізовані в повному обсязі. Система забезпечує адаптивну вибірку завдань з бази даних відповідно до рівня знань користувача, що підтверджується коректною роботою для різних тестових акаунтів. Реалізовано механізм миттєвого зворотного зв'язку у вправах з візуальною індикацією правильних та неправильних відповідей, а також система нарахування балів. Модуль персонального словника дозволяє користувачам зберігати нові слова, а інтегрований чат-бот коректно обробляє запити, генерує відповіді та унікальні завдання на задану тему, дотримуючись інструкцій щодо мови та стилю спілкування.

5.2 Оцінка якості роботи системи

Продуктивність системи є задовільною: час відповіді сервера на запити для завантаження вправ в середньому не перевищує 300 мс, що відповідає поставленим вимогам. Час відповіді AI-асистента залежить від завантаженості зовнішнього сервісу Google Gemini API, але під час тестування в середньому складав 2-4 секунди, що є прийнятним для інтерактивного діалогу. Надійність системи підтверджується коректною обробкою помилкових ситуацій, таких як відсутність токена автентифікації або відсутність питань для певного рівня, у таких випадках користувач отримує відповідне інформаційне повідомлення.

5.3 Аналіз ефективності прийнятих рішень

Ефективність проєкту значною мірою залежала від правильності вибору технологічного набору та архітектурних підходів. Обрана клієнт-серверна архітектура виправдала себе, дозволивши чітко розмежувати логіку та спростити процес розробки. Використання бібліотеки React на клієнтській стороні забезпечило створення швидкого та відзивчивого інтерфейсу. На серверній стороні Node.js у зв'язці з Express продемонстрував високу ефективність в обробці асинхронних запитів до бази даних та зовнішніх API. Інтеграція ORM Sequelize значно спростила роботу з базою даних SQLite. Вибір Google Gemini API як AI-сервісу виявився вдалим рішенням, оскільки він надає потужний функціонал генерації тексту.

5.4 Обмеження та недоліки

Незважаючи на успішну реалізацію основного функціонала, проєкт має певні обмеження та недоліки, які відкривають можливості для подальшого розвитку. По-перше, поточна база питань для вправ є обмеженою і вимагає постійного наповнення для забезпечення різноманітності завдань. По-друге, якість відповідей та вправ, згенерованих AI, хоча і є високою, не завжди може бути ідеальною. По-третє, відсутня повноцінна мобільна версія додатку, хоча веб-інтерфейс є адаптивним.

ВИСНОВКИ

В рамках виконаної роботи було успішно спроєктовано та розроблено інтерактивний веб-додаток для вивчення англійської мови з використанням технологій штучного інтелекту. Було проведено детальний аналіз предметної галузі, на основі якого сформульовано вимоги до системи та обрано оптимальний технологічний набір.

Основним результатом роботи є працездатний програмний продукт, що реалізує клієнт-серверну архітектуру. Серверна частина, розроблена на платформі Node.js з використанням фреймворку Express та ORM Sequelize, забезпечує надійну бізнес-логіку, включаючи автентифікацію користувачів за допомогою JWT та взаємодію з базою даних SQLite. Клієнтська частина, створена на бібліотеці React, надає користувачам інтуїтивно зрозумілий та відзивчивий інтерфейс.

Було успішно реалізовано ключові функціональні модулі, що складають ядро додатку. Розроблено три типи інтерактивних вправ ("Вибір відповіді", "Побудова речення", "Переклад слів"), які адаптуються до рівня знань користувача, що зберігається у його профілі. Система надає миттєвий зворотний зв'язок та веде підрахунок балів, що сприяє підвищенню мотивації. Центральним елементом є модуль "AI Language Buddy", який інтегровано із зовнішнім сервісом Google Gemini API. Він дозволяє користувачам вести вільний текстовий діалог, отримувати відповіді на питання та генерувати унікальні вправи на обрану тему, що забезпечує гнучкість та нескінченну варіативність навчального процесу.

Проведене тестування підтвердило відповідність розробленої системи сформульованим функціональним та нефункціональним вимогам. Система демонструє стабільну роботу, коректну обробку даних та прийнятну продуктивність.

Таким чином, розроблений сучасний та функціональний прототип освітнього веб-додатку, що підтвердило ефективність поєднання структурованих вправ та інтерактивної взаємодії зі штучним інтелектом. Проєкт є повністю

функціональним, має значний потенціал для подальшого розвитку та може слугувати основою для створення комерційного продукту.

Вихідний код проєкту доступний у GitHub репозиторії за посиланням: https://github.com/NureMiachKateryna/2025_B_KKP_PZPI-22-9_Miach_K_O

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Language Models are Few-Shot Learners. *arXiv.org*. URL: https://arxiv.org/abs/2005.14165 (дата звернення: 10.06.2025).

2. Gemini API | Google AI for Developers. *Google AI for Developers*. URL: https://ai.google.dev/gemini-api/docs (дата звернення: 13.06.2025).

3. Фрімен Е., Робсон Е., Сьєрра К., Бейтс Б. Head First. Патерни проєктування. 2-е вид. / пер. з англ. Київ : Фабула, 2021. 672 с.

4. Бенкс А., Порселло Є. React і Redux: функціональна веб-розробка. / пер. з англ. Харків : Фабула, 2019. 320 с.

5. Express - Node.js web application framework. *Express - Node.js web application framework*. URL: https://expressjs.com/ (дата звернення: 11.06.2025).

6. Sequelize. *Sequelize | Feature-rich ORM for modern TypeScript & JavaScript*. URL: https://sequelize.org/ (дата звернення: 11.06.2025).

7. Річардс М. Основи програмної архітектури: інженерний підхід / пер. з англ. Київ : Ранок, 2023. 416 с.

8. Мартін Р. Чиста архітектура. Мистецтво розробки програмного забезпечення. / пер. з англ. Харків : Фабула, 2018. 368 с.

9. Godwin-Jones, R. Personal learning environments. *Language Learning & Technology*. URL: https://www.lltjournal.org/item/10125-73489 (дата звернення: 12.06.2025).

10. JSON Web Token Introduction - jwt.io. *JSON Web Tokens - jwt.io*. URL: https://jwt.io/introduction (дата звернення: 12.06.2025).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

ДОДАТОК Б

Специфікація ПЗ

CS330 Software Engineering

Software Requirements Specification (SRS) Template

Items that are intended to stay in as part of your document are in **bold**; explanatory comments are in *italic* text. Plain text is used where you might insert wording about your project.

The document in this file is an annotated outline for specifying software requirements, adapted from the IEEE Guide to Software Requirements Specifications (Std 830-1993).

Tailor this to your needs, removing explanatory comments as you go along. Where you decide to omit a section, keep the header, but insert a comment saying why you omit the data.

Software system for teaching foreign words

SmartCore Developers

Miach Kateryna

Viktoriia Serdiuk

Software Requirements Specification

Document

Version: 1.0 **Date: 17.06.2025**

Зміст

1 ВСТУП

1.1 Огляд продукту

У межах цього проєкту буде створено інтерактивний веб-застосунок для вивчення англійської мови. Застосунок працюватиме через веб-браузер і буде орієнтований на користувачів, які бажають покращити свій словниковий запас, граматику та навички спілкування. Однією з ключових особливостей системи стане впровадження чат-бота, що базуватиметься на технологіях штучного інтелекту, який зможе вести діалоги з користувачем, ставити запитання, виправляти помилки та мотивувати до навчання. Застосунок матиме низку функціональних сторінок: реєстрацію, вхід, профіль, словник, сторінку міні-тестів, сторінки вправ: вибір правильної відповіді та побудова речення, а також взаємодію з чат-ботом. Передбачається, що користувач зможе проходити короткі тести для самоперевірки, переглядати вивчені слова, а також отримувати зворотний зв’язок від AI. Реалізація продукту планується з використанням такого стека технологій:

- Frontend: HTML, CSS, JavaScript, React;
- Backend: Node.js з Express.js;
- База даних: Sequelize ORM з SQLite.

1.2 Мета

Головна мета – розробити ефективний онлайн-інструмент для самостійного вивчення англійської мови з використанням інноваційних підходів до навчання. Продукт має забезпечити зручність, доступність та гнучкість у навчальному процесі, завдяки можливості навчатися у будь-який час і з будь-якого пристрою. Система буде спрямована на: адаптацію навчального матеріалу до рівня користувача; інтеграцію чат-бота, який допомагатиме у вивченні мови в режимі реального часу, генеруватиме відповіді, ставитиме питання та надаватиме індивідуальні рекомендації.

1.3 Межі

На першому етапі розробки застосунок матиме обмежений функціонал, орієнтований на основні потреби користувача, а саме:

- підтримка лише англійської мови;
- відсутність повноцінної адміністративної панелі для керування контентом;
- база даних SQLite, яка обмежена у продуктивності при високому навантаженні;
- історія проходження тестів не зберігатиметься;
- профіль користувача не міститиме повної статистики про активність;
- використання локального середовища для запуску.

1.4 Посилання

Для розробки проєкту можна скористатися такими офіційними джерелами:
React – бібліотека для створення інтерфейсів користувача (https://react.dev/)
Node.js – серверна платформа для JavaScript (https://nodejs.org/)
Express.js – фреймворк для створення веб-серверів (https://expressjs.com/)
Sequelize ORM – ORM для роботи з базами даних (https://sequelize.org/)
SQLite – легка вбудована реляційна база даних (https://sqlite.org/)
Google Gemini – платформа штучного інтелекту від Google (https://gemini.google.com/)

1.5 Означення та абревіатури

AI (Artificial Intelligence) – Штучний інтелект
JWT (JSON Web Token) – формат для безпечної передачі токенів авторизації
ORM (Object-Relational Mapping) – технологія для взаємодії з базами даних у вигляді об’єктів
SQLite Легка реляційна база даних, що не потребує окремого сервера
GitHub – Платформа для спільної розробки та зберігання коду

2 ЗАГАЛЬНИЙ ОПИС

2.1 Перспективи продукту

Проєктований інтерактивний веб-додаток буде самостійним програмним продуктом, призначеним для кінцевих користувачів, що вивчають англійську мову. Він не є частиною більшої системи, проте його архітектура має передбачати можливість майбутньої інтеграції з іншими освітніми платформами або розширення функціонала.
Продукт буде позиціонуватися на ринку як комплексна екосистема для самостійного навчання, що охоплює всі етапи: від визначення рівня знань до щоденної практики. У майбутньому перспективи продукту включатимуть розробку повноцінних мобільних додатків, розширення бази мов для вивчення та інтеграцію з системами управління навчанням.

2.2 Функції продукту

Система буде надавати користувачам наступний набір основних функцій:

- головна сторінка: виконує роль інформаційного лендингу, що презентує основні можливості та переваги додатку;

- реєстрація користувача: користувачі можуть створювати новий обліковий запис, надаючи свої дані;
- автентифікація: зареєстровані користувачі можуть входити до системи за допомогою email та пароля;
- профіль користувача: доступ до особистого профілю з можливістю редагування даних і рівня складності;
- вибір вправ: сторінка з вибором трьох типів інтерактивних вправ – «вибір правильної відповіді», «побудова речення» та «переклад слів»;
- проходження адаптивних вправ: автоматичний підбір завдань відповідно до рівня користувача;
- інтерактивний ai-чат: можливість вести текстовий діалог з ai-асистентом на вільну або обрану тему;
- ведення персонального словника: можливість додавати нові слова до особистого словника під час виконання вправ.

2.3 Характеристики користувачів

Продукт орієнтований на широку аудиторію користувачів, які прагнуть вивчити або покращити англійську мову. Система розрахована на два основні типи користувачів: нового користувача, який може ознайомитися з головною сторінкою, та зареєстрованого користувача, якому доступний весь функціонал. Програмне забезпечення буде підтримувати учнів з різним рівнем підготовки, від початкового до просунутого, адаптуючи контент відповідно до вказаного ними рівня. Для взаємодії з системою від користувача очікується лише базовий рівень комп'ютерної грамотності, що включає вміння користуватися веб-браузером. Основною мотивацією цільової аудиторії є бажання вивчити мову "з нуля", покращити існуючі навички для кар'єрного росту або підготуватися до подорожей.

2.4 Загальні обмеження

При розробці та функціонуванні системи необхідно враховувати низку загальних обмежень. Перш за все, існують технологічні обмеження, оскільки продукт є веб-додатком і вимагатиме від користувача наявності пристрою з доступом до мережі Інтернет та сучасного веб-браузера. Важливим обмеженням є залежність від зовнішніх сервісів, адже функціонал AI чат-бота буде повністю залежати від стабільності та умов надання сервісу Google Gemini API. Також слід враховувати обмеження обраної бази даних: СУБД SQLite, хоча і є зручною для розробки, може мати обмеження продуктивності при високому навантаженні. Нарешті, на поточному етапі існують мовні обмеження, оскільки система призначена виключно для вивчення англійської мови україномовними користувачами.

2.5 Припущення й залежності

Проєктування системи базується на кількох ключових припущеннях. Припускається, що користувач матиме стабільне інтернет-з'єднання, необхідне для коректної роботи всіх мережевих функцій, особливо взаємодії з AI-асистентом. Система має низку залежностей, що впливають на її функціонування. Проєкт критично залежить від стабільності роботи зовнішнього сервісу Google Gemini API. Крім того, він залежить від сторонніх програмних бібліотек та пакетів (npm-пакетів), таких як React, Express та Sequelize, майбутні оновлення яких можуть вимагати адаптації кодової бази. Доступність додатку для кінцевих користувачів буде залежати від надійності та стабільності роботи обраного хостинг-провайдера.

3. КОНКРЕТНІ ВИМОГИ

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

Інтерфейс користувача повинен бути розроблений з урахуванням принципів UX/UI, забезпечуючи легкість навігації, приємний дизайн і зручність користування. Він має бути адаптивним і коректно відображатися на різних типах пристроїв та в різних браузерах. Основні компоненти інтерфейсу включають:

- Форми реєстрації та авторизації із валідацією введених даних та повідомленнями про помилки;
- Особистий кабінет користувача з можливістю перегляду та редагування персональної інформації та зміни пароля;
- Сторінки словника для додавання, видалення та повторення слів;
- Сторінка з міні-тестами із таймером, підрахунком балів та відображенням результатів;
- Сторінки вправ, що включають вибір правильної відповіді та побудову речення, що допомагають закріпити граматичні та лексичні навички;
- Чат-бот на базі AI для інтерактивної взаємодії та навчання користувача.

3.1.2 Апаратний інтерфейс

Програмний продукт буде працювати на стандартних пристроях з операційними системами Windows, macOS, Linux. Не передбачається використання спеціального обладнання. Всі необхідні обчислювальні ресурси для роботи обробляються на сервері, клієнтською частиною є веб-браузер. Мінімальні апаратні вимоги для клієнтського пристрою:

- Процесор з тактовою частотою не менше 1.5 ГГц;
- Оперативна пам’ять – не менше 2 ГБ;

- Доступ до стабільного інтернет-з’єднання для взаємодії з сервером.

3.1.3 Програмний інтерфейс

Програмне забезпечення буде реалізоване за принципом клієнт-серверної архітектури. Клієнтська частина створюватиметься з використанням бібліотеки React, що дає змогу розробити динамічний і зручний інтерфейс користувача. Серверна частина працюватиме на платформі Node.js із фреймворком Express.js, який дозволяє ефективно обробляти HTTP-запити та легко масштабувати систему. Обмін даними між клієнтом і сервером відбуватиметься через REST API, що забезпечить чітку структуру запитів і стабільну взаємодію компонентів. Формат передавання даних — JSON, який є загальноприйнятим і зручним для веб-розробки. Для роботи з базою даних буде використано ORM Sequelize, який спрощує доступ до даних завдяки об’єктно-орієнтованому підходу. Також планується використання бази даних SQLite.

3.1.4 Комунікаційний протокол

Програмне забезпечення буде реалізовано за принципом взаємодії клієнтської та серверної частин через протокол HTTP або HTTPS. Для обміну даними між клієнтом і сервером використовуватиметься REST API з передачею інформації у форматі JSON. Планується впровадження механізму автентифікації користувачів за допомогою токенів (JWT), що дозволить забезпечити безпечний доступ до системи без повторного входу. З метою захисту персональних даних паролі користувачів будуть хешуватися за допомогою алгоритму bcrypt перед збереженням у базі даних. У фінальній версії, при розгортанні в мережі Інтернет, система використовуватиме безпечний протокол HTTPS для шифрування всього трафіку між клієнтом і сервером.

3.1.5 Обмеження пам’яті

Програма орієнтована на роботу на сучасних пристроях з мінімальними апаратними вимогами, що включають не менше 2 ГБ оперативної пам’яті на клієнтському пристрої. Серверна частина буде розгорнута з урахуванням обмежень доступної пам’яті на хостингу або сервері, з можливістю масштабування при збільшенні кількості користувачів. Для бази даних на початковому етапі планується використання SQLite, що має невеликі вимоги до ресурсів, з подальшим переходом на більш потужні СУБД, наприклад, MySQL чи PostgreSQL.

3.1.6 Операції

У процесі взаємодії з системою користувач зможе здійснювати такі дії:

- Зареєструвати новий обліковий запис, ввівши необхідні дані;

- Авторизуватися в системі з використанням електронної пошти та пароля;
- Переглядати, редагувати свій профіль, змінювати пароль або видаляти обліковий запис;
- Переглядати сторінку словника з можливістю перекладу та вивчення слів;
- Проходити інтерактивні вправи, такі як: вибір правильної відповіді, переклад слова, побудова речення;
- Отримувати результати виконання вправ;
- Користуватися функціями штучного інтелекту для покращення навчання.

3.1.7 Функції продукту

Програмний продукт повинен забезпечити реалізацію таких основних функцій:

- Реєстрація нових користувачів та вхід до особистого кабінету;
- Керування профілем: редагування особистих даних, зміна пароля, видалення облікового запису;
- Доступ до модуля словника для перегляду, вивчення та перекладу лексики;
- Проходження інтерактивних вправ, зокрема: вибір правильної відповіді, переклад слова, побудова речень;
- Обробка відповідей користувача та надання зворотного зв’язку щодо правильності;
- Взаємодія між клієнтською та серверною частинами через REST API з обміном даними у форматі JSON;
- Інтеграція з AI-сервісом (Google Gemini) для адаптації навчального контенту та підвищення ефективності навчання.

3.1.8 Припущення й залежності

- Користувач має доступ до сучасного веб-браузера, що підтримує JavaScript та актуальні вебтехнології;
- Пристрій користувача відповідає мінімальним технічним вимогам, необхідним для стабільної роботи вебзастосунку;
- Забезпечено стабільне інтернет-з’єднання для коректної взаємодії клієнтської частини з сервером у режимі реального часу;
- На етапі розробки база даних реалізується на SQLite;
- Програмне забезпечення включає інтеграцію з AI-сервісами для реалізації інтелектуальних функцій: адаптації матеріалу до рівня користувача, надання розумних підказок тощо;

- Усі взаємодії з системою (реєстрація, вхід, виконання вправ) реалізуються через REST API з використанням формату JSON та механізмів автентифікації.

3.2 Властивості програмного продукту

Програмний продукт, що розробляється, повинен володіти низкою ключових властивостей, які визначають його якість та відповідність очікуванням користувачів. По-перше, система повинна бути гнучкою та адаптивною. Це означає, що навчальний контент, зокрема вправи та складність діалогів з AI-асистентом, має динамічно підлаштовуватися під рівень знань, вказаний користувачем. По-друге, продукт має бути інтерактивним та залучаючим. Взаємодія з системою не повинна зводитися до пасивного споживання інформації. Користувач повинен активно брати участь у процесі: виконувати різноманітні завдання, вести діалог, самостійно формулювати відповіді та отримувати миттєвий зворотний зв'язок. По-третє, система повинна бути надійною та стабільною. Це передбачає коректну обробку помилок, таких як втрата інтернет-з'єднання або недоступність зовнішніх сервісів, та надання користувачеві зрозумілих повідомлень про стан системи. По-четверте, продукт має бути розширюваним. Архітектура повинна дозволяти в майбутньому легко додавати нові типи вправ, нові теми для обговорення в чаті та підключати інші мови для вивчення без необхідності кардинальної перебудови всієї системи.

3.3 Атрибути програмного продукту

3.3.1 Надійність

Система повинна залишатися працездатною при виникненні непередбачуваних ситуацій. Вона має коректно обробляти помилки, що надходять від зовнішнього Google Gemini API, і відображати користувачеві відповідне повідомлення, не перериваючи роботу всього додатку. У випадку помилок при збереженні даних у базу, система повинна інформувати про це користувача, а не завершуватися з критичною помилкою.

3.3.2 Доступність

Веб-додаток повинен бути доступним для користувачів 24 години на добу, 7 днів на тиждень, за винятком планових технічних робіт. Очікуваний показник доступності для продуктивного середовища має становити не менше 99.5%.

3.3.3 Безпека

Безпека є критично важливим атрибутом. Система повинна забезпечувати:

- конфіденційність даних користувача: всі дані, що передаються між клієнтом та сервером, повинні шифруватися за допомогою протоколу HTTPS;
- безпечне зберігання паролів: паролі користувачів не повинні зберігатися у відкритому вигляді, вони мають хешуватися за допомогою надійного алгоритму (наприклад, bcrypt);
- автентифікація та авторизація: доступ до персональних даних (словник, профіль) та захищених функцій (проходження вправ, чат) має надаватися лише автентифікованим користувачам, для цього використовується механізм на основі JWT (JSON Web Tokens);

3.3.4 Супроводжуваність

Кодова база проєкту має бути структурованою та добре документованою. Архітектурне рішення з чітким розділенням на клієнтську та серверну частини, а також модульна структура на обох сторонах (компоненти в React, маршрутизатори в Express) спрощують внесення змін, виправлення помилок та додавання нового функціонала в майбутньому.

3.3.5 Переносимість

Серверна частина, розроблена на Node.js, є кросплатформною і може бути розгорнута на будь-якій операційній системі (Windows, macOS, Linux). Клієнтська частина є веб-додатком і вимагає лише наявності сучасного веб-браузера, що робить її незалежною від платформи користувача.

3.3.6 Продуктивність

Система повинна забезпечувати комфортний час відгуку. Час завантаження основних сторінок не повинен перевищувати 3 секунди на середньостатистичному інтернет-з'єднанні. Відповідь сервера на запити клієнта (крім запитів до AI) не повинна перевищувати 500 мілісекунд. Час відповіді від AI-чату залежить від зовнішнього сервісу, але очікується в межах 2-5 секунд.

3.4 Вимоги бази даних

Перш за все, система повинна гарантувати цілісність даних. Це досягається шляхом використання механізмів первинних (Primary Key) та зовнішніх (Foreign Key) ключів у рамках реляційної моделі.

Другою вимогою є структуроване зберігання. База даних повинна мати чітко визначену схему, що складається з основних таблиць, які відповідають ключовим сутностям предметної області. Поля цих таблиць повинні використовувати відповідні типи даних, такі як INTEGER, VARCHAR, TEXT та DATETIME, для забезпечення коректності та типізації збереженої інформації, що є важливим для подальшої роботи ORM Sequelize.

Третя вимога – ефективність виконання запитів. Структура таблиць та їх індексація, зокрема автоматичне індексування первинних та унікальних ключів (наприклад, id та email у таблиці Users), повинні забезпечувати швидке виконання основних операцій. До таких операцій належать: вибірка користувача за ідентифікатором, пошук питань за рівнем та типом, отримання персонального словника, а також операції вставки та оновлення даних.

Четвертою вимогою є масштабованість. Хоча на початковому етапі для спрощення розробки та розгортання використовується файлова СУБД SQLite, спроєктована схема даних є універсальною. Завдяки використанню ORM Sequelize, система спроєктована таким чином, щоб у майбутньому можна було легко мігрувати на більш потужні клієнт-серверні СУБД. Вимоги до бази даних мають охоплювати всю систему, а не тільки такі як PostgreSQL або MySQL, без значних змін у бізнес-логіці додатку.

П'ятою вимогою є забезпечення можливості резервного копіювання. Повинна бути передбачена процедура для створення регулярних резервних копій файлу бази даних, що є критично важливим для запобігання втраті даних у разі апаратних збоїв чи інших непередбачуваних ситуацій.

3.5 Інші вимоги

Усі компоненти проєкту мають відповідати встановленим технічним вимогам. Програмний код повинен бути написаний згідно зі стандартами стилю для відповідної мови, бути структурованим, читабельним і містити коментарі в складних місцях для полегшення підтримки. Розробка має вестися з використанням системи контролю версій Git, а також передбачати наявність віддаленого репозиторію для командної роботи, резервного копіювання та відстеження змін. Серверна частина повинна реалізовувати логування ключових подій та помилок, що дозволить ефективно моніторити систему й вчасно реагувати на збої. Оскільки проєкт використовує зовнішній сервіс Google Gemini API, необхідно подбати про безпечне зберігання API-ключа через файл .env, не допускаючи його потрапляння у відкритий клієнтський код.

ДОДАТОК В

Слайди презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Інтерактивний веб-додаток для навчання англійської мови із чат-ботом на базі AI

ПІБ, група:
М’яч Катерина Олександрівна
ПЗПІ-22-9

Керівник:
ст.викл. кафедри ПІ Катерина Зибіна

17 червня 2025

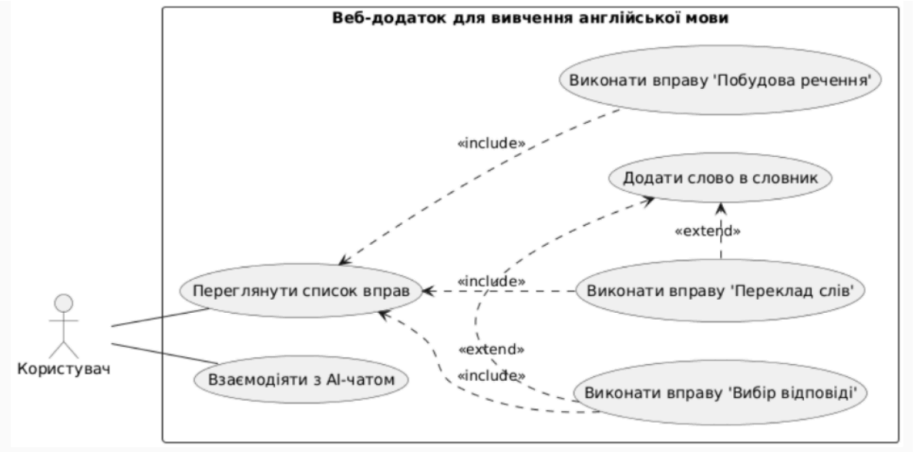
Мета роботи

- Аналіз предметної області онлайн-освіти та існуючих AI-рішень для вивчення мов
- Вивчення особливостей роботи чат-ботів на базі штучного інтелекту
- Розробка архітектури клієнт-серверного веб-додатку для мовної практики
- Реалізація функціональності: вправи, словник, AI-чат на основі Google Gemini API
- Проєктування користувацького інтерфейсу та UX
- Тестування системи та аналіз отриманих результатів

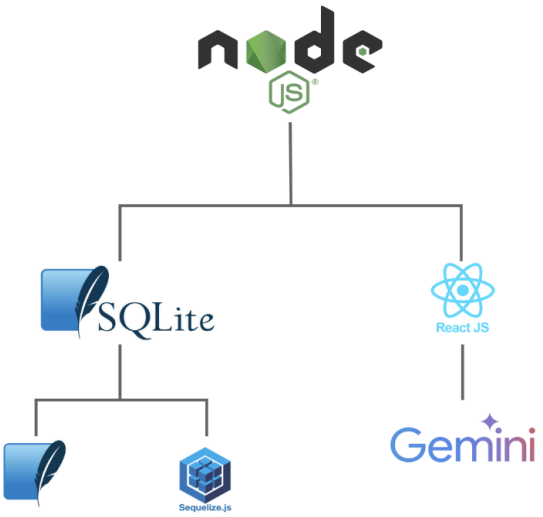
Аналіз проблеми

Назва платформи	Переваги	Недоліки
Duolingo	Гейміфікація, велика база користувачів, безкоштовність	Відсутність глибокої граматики, обмежена практика письма
Babbel	Структуровані уроки, реальні діалоги	Платна модель, обмежений вибір тем
Memrise	Слова в реальному контексті, відео з носіями	Слабкий розвиток граматики, відсутність чат-функціоналу
Busuu	Соціальні функції, перевірка носіями	Залежність від активності користувачів, затримки у перевірці
ChatGPT	Гнучкість, можливість діалогу на будь-яку тему	Відсутність структури, немає системи збереження прогресу

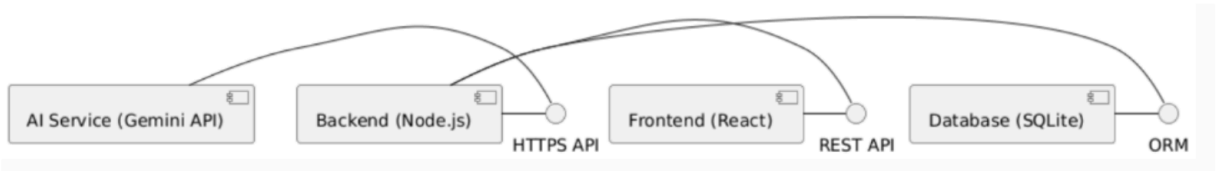
Постановка задачі та опис системи



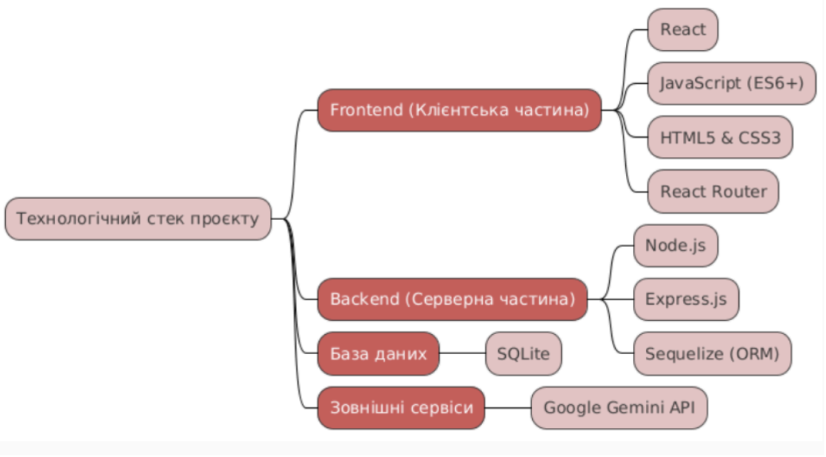
Вибір технологій розробки



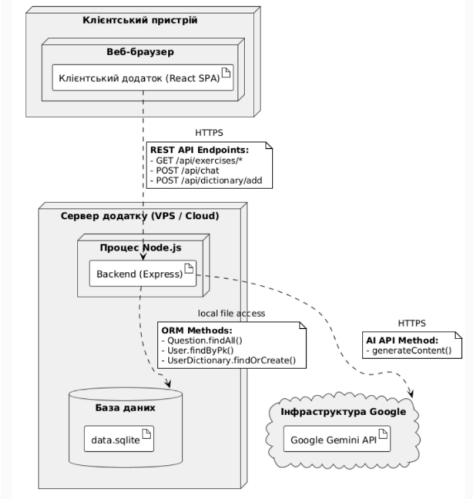
Архітектура створеного програмного забезпечення



Опис програмного забезпечення, що було використано у дослідженні



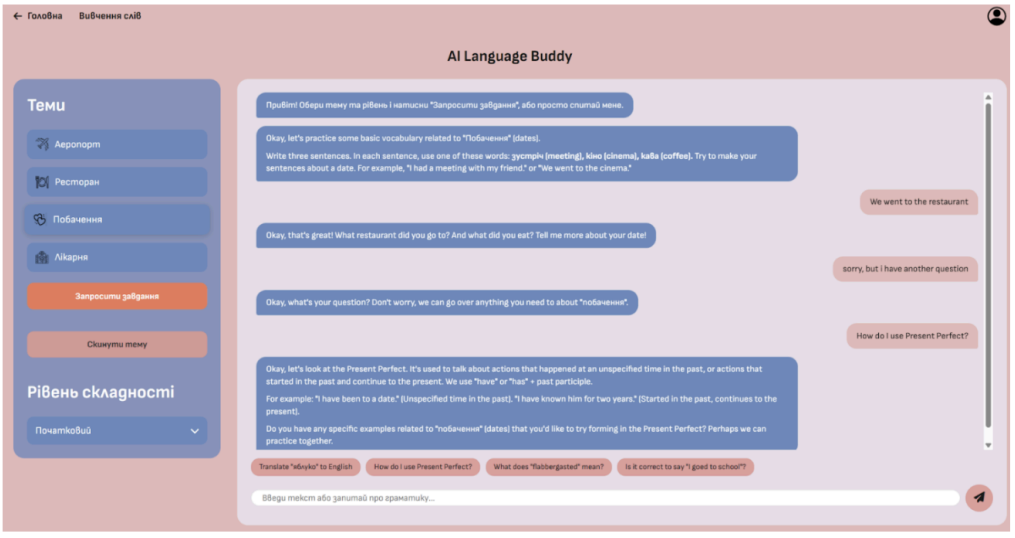
Дизайн системи



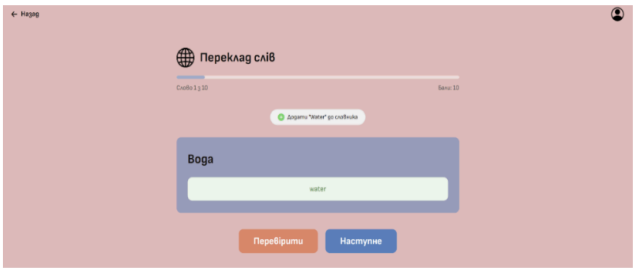
Приклад реалізації

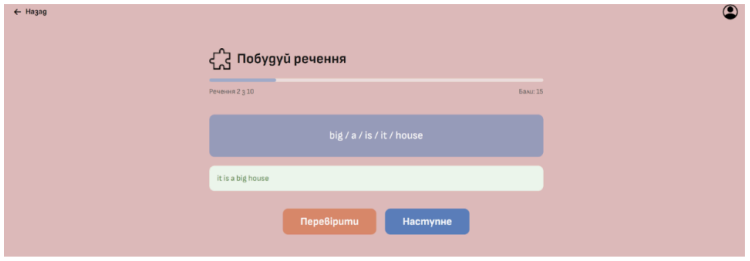
```
// Створюємо інструкцію для AI
let instruction = `(Tutor Mode: Respond ONLY in English. The user's level is ${user.level}. The topic
is ${topic || 'general conversation'}.`;

// Додаємо спеціальну інструкцію, якщо це запит на завдання
if (message.includes("Please give me a simple language exercise")) {
    instruction += ` The user requested an exercise. Provide a simple language task.)`;
} else {
    instruction += ` Just continue the conversation naturally.)`;
}

// Формуємо фінальний промпт для відправлення
const finalMessageToSend = `${instruction}\n\nUser message: "${message}"`;

// Відправляємо в AI
const result = await chat.sendMessage(finalMessageToSend);
```


Інтерфейс користувача



Інтерфейс користувача





Тестування

ID	Опис	Передумови	Кроки тесту	Очікуваний результат	Пріоритет	Результат тесту	Середовище	Дата створення
1	Перевірка правильної відповіді у вправі	- Користувач авторизований. - Користувач знаходиться на сторінці вправи "Вибір відповіді". - На екрані відображено питання та варіанти.	К: Обирає правильний варіант відповіді. С: Перевіряє відповідь, нараховує бали, змінює стан isCorrect на true. К: Перевіряє, що кнопка підсвітилась зеленим, а рахунок балів збільшився.	Кнопка підсвічена зеленим, рахунок балів оновлено, з'явилась кнопка "Додати в словник".	High	PASS	Browser: Chrome OS: Windows 11	12.06.2024
2	Генерація вправи від AI за темою	- Користувач авторизований. - Користувач знаходиться на сторінці AI-чату. - Обрано тему "Ресторан" та рівень "Середній".	К: Натискає кнопку "Запросити завдання". 2. С: Відправляє запит на /api/chat зі спеціальним промптом. 3. С: Отримує відповідь від AI. 4. К: Перевіряє, що у чаті з'явилось нове повідомлення з вправою на тему ресторану.	У вікні чату з'явилося релевантне завдання на тему ресторану, написане англійською мовою.	High	PASS	Browser: Chrome OS: Windows 11	12.06.2024
3	Додавання слова до словника	- Користувач авторизований. - Користувач пройшов питання у вправі. - Відображається кнопка "Додати в словник".	К: Натискає кнопку "Додати ... до словника". С: Відправляє POST-запит на /api/dictionary/add з даними слова. К: Перевіряє, що з'явилось сповіщення про успішне додавання.	З'являється alert з повідомленням "Слово успішно додано!".	High	PASS	Browser: Chrome OS: Windows 11	12.06.2024

Підсумки

У результаті роботи було створено інтерактивний веб-додаток, який є реалістичним та корисним інструментом для вивчення англійської мови. Система може використовуватися для самостійного навчання, поєднуючи вправи та спілкування з AI. Можливий розвиток проєкту включає розробку мобільних застосунків та додавання нових функцій, таких як практика вимови.