МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ЗАСТОСУНКУ ДЛЯ ВИВЧЕННЯ ІНОЗЕМНОЇ МОВИ З ІНТЕГРАЦІЄЮ ГЕНЕРАТИВНИХ МОДЕЛЕЙ»**

Виконав: студент 4 курсу, групи 6.1212-пі-2
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
В.М. Вишневецький
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, к.ф.-м.н. Кривохата А.Г.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент зав. кафедри комп'ютерних наук, доцент, к.пед.н. Пшенична О.С.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Математичний

Кафедра програмної інженерії

Рівень вищої освіти Бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Вишневецькому Володимиру Максимовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка застосунку для вивчення іноземної мови з інтеграцією генеративних моделей

керівник роботи Кривохата Анастасія Григорівна, к.ф.-м.н.
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 02 » грудня 2025 року № 1741-с

2. Строк подання студентом роботи 01.06.2026 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз предметної області та програмних засобів.
2. Проєктування інформаційної системи.
3. Програмна реалізація та інтеграція генеративних моделей.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 09.12.2025 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	19.12.2025	
2.	Збір вихідних даних.	16.01.2026	
3.	Обробка методичних та теоретичних	12.02.2026	
	джерел.		
4.	Розробка першого та другого розділу.	20.03.2026	
5.	Розробка третього розділу.	15.05.2026	
6.	Оформлення та нормоконтроль	29.05.2026	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	16.06.2026	

Студент _____ В.М. Вишневецький
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.Г. Кривохата
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка застосунку для вивчення іноземної мови з інтеграцією генеративних моделей»: 48 с., 31 рис., 1 табл., 18 джерел, 1 додаток.

АДАПТИВНЕ НАВЧАННЯ, ГЕНЕРАТИВНІ МОДЕЛІ, ВЕЛИКІ МОВНІ МОДЕЛІ, МАШИННЕ НАВЧАННЯ, ОСВІТНІ ВЕБЗАСТОСУНКИ, ПЕРСОНАЛІЗАЦІЯ НАВЧАННЯ, ПРОМПТ-ІНЖЕНЕРІЯ, ШТУЧНИЙ ІНТЕЛЕКТ.

Об’єкт дослідження – процес розробки платформи для вивчення іноземних мов.

Мета роботи: розробка застосунку для вивчення іноземної мови з інтеграцією генеративних моделей.

Методи дослідження – методи об’єктно-орієнтованого програмування, методи машинного навчання, методи використання великих мовних моделей (LLM) для генерації, аналізу й адаптації навчального контенту.

У межах кваліфікаційної роботи було розроблено вебзастосунок для вивчення іноземної мови з використанням генеративних моделей штучного інтелекту. Для реалізації поставлених завдань було використано мову програмування Python. Як вебсерверну складову застосовано фреймворк FastAPI.

Новизна підходу полягає у використанні великих мовних моделей як основного компонента, що забезпечує адаптивне навчання відповідно до цілей студента. Щоб зменшити частку галюцинацій у відповідях, зокрема під час генерації навчального контенту, зворотного зв’язку та навчальних модулів, було застосовано техніки промпт-інженерії та генерацію з доповненням через пошук (RAG), реалізовану на основі архітектури векторної бази даних Weaviate.

SUMMARY

Bachelor’s qualifying paper “Development of an Application for Foreign Language Learning Integrating Generative AI Models”: 48 pages, 31 figures, 1 table, 18 references, 1 supplement.

ADAPTIVE LEARNING, GENERATIVE MODELS, LARGE LANGUAGE MODELS, MACHINE LEARNING, EDUCATIONAL WEB APPLICATIONS, LEARNING PERSONALIZATION, PROMPT ENGINEERING, ARTIFICIAL INTELLIGENCE.

The object of the study is the process of developing a platform for learning foreign languages.

The aim of the study is to develop an application for learning a foreign language with the integration of generative models.

The methods of research include object-oriented programming methods, machine learning methods, as well as methods of using large language models (LLM) for the generation, analysis, and adaptation of educational content.

Within the scope of the bachelor’s qualification paper, a web application for learning a foreign language using generative artificial intelligence models was developed. The Python programming language was used to implement the defined tasks. The FastAPI framework was applied as the web server component of the system.

The novelty of the approach lies in the use of large language models as a core component, enabling adaptive learning aligned with individual student goals. To reduce the occurrence of hallucinations in model responses, particularly during the generation of educational content, feedback, and learning modules, prompt engineering techniques and Retrieval-Augmented Generation (RAG) were applied, implemented using the Weaviate vector database architecture.

ЗМІСТ

Завдання на кваліфікаційну роботу……..2
Реферат 4
Summary 5
Вступ..... 7
1 Визначення вимог до системи вивчення іноземної мови з інтеграцією генеративних моделей 9
1.1 Характеристика інформаційної системи 9
1.2 Переваги використання мовних моделей для генерації навчального контенту 10
1.3 Програмне забезпечення розробки системи..... 11
1.4 Вибір великої мовної моделі..... 12
1.5 Технічне завдання 14
2 Проєктування системи вивчення іноземної мови з інтеграцією генеративних моделей..... 17
2.1 Розробка концептуальної моделі..... 17
2.2 Побудова логічної структури..... 20
2.3 Моделювання фізичних компонентів системи 23
3 Програмна реалізація системи вивчення іноземної мови 26
3.1 Реалізація інтерфейсу 26
3.2 Промпт-інженерія 30
3.3 Розробка векторної бази даних..... 32
3.4 Моніторинг застосунку 34
3.5 Розробка сторінки технічної підтримки 36
Висновки 40
Перелік посилань..... 42
Додаток А Модуль промпт-інженерії 45

ВСТУП

Сучасний розвиток IT індустрії призводить до того, що ринок вимагає інноваційних рішень, здатних забезпечити персоналізацію та інтерактивність з користувачем, зокрема в галузі освіти. Це говорить про необхідність розробки систем, які надають можливість обробки та аналізу великих обсягів даних та в режимі реального часу генерують релевантний контент для користувача.

Завдяки використанню можливостей великих мовних моделей, користувачі мають змогу отримати щоденну мовну практику, що охоплює читання, аудіювання та спілкування. Дані аспекти забезпечують новий рівень навчального досвіду, що обумовлює актуальність даного проєкту.

Мета – розробка застосунку для вивчення іноземної мови з інтеграцією генеративних моделей.

Для досягнення поставленої мети необхідно вирішити такі задачі:

- проаналізувати програмне забезпечення: бібліотеки та фреймворки для використання великих мовних моделей, RAG-систем, машинного навчання та створення вебзастосунків;
- дослідити предметну область вивчення іноземних мов із використанням технологій штучного інтелекту (ШІ);
- спроєктувати архітектуру застосунка;
- розробити застосунок для вивчення іноземних мов з інтеграцією генеративних моделей;
- створити систему моніторингу розробленого застосунку з використанням фреймворку Phoenix;
- провести тестування інформаційної системи.

Об’єкт дослідження – процес розробки платформи для вивчення іноземних мов.

Предмет дослідження – методи та засоби інтеграції LLM у застосунки на основі використання фреймворку FastAPI, реляційної бази даних SQLite3,

векторної бази даних Weaviate для побудови RAG-системи та платформи Arize Phoenix для моніторингу й аналізу продуктивності генеративних моделей.

Методи дослідження – методи об’єктно-орієнтованого програмування, методи машинного навчання, методи використання великих мовних моделей для генерації, аналізу й адаптації навчального контенту.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, переліку посилань та 1 додатку. Загальний обсяг роботи становить 48 сторінок. Додатки викладено на 4 сторінках. У переліку посилань 18 використаних джерел.

Перший розділ присвячений аналізу предметної області, аналогічних рішень, недоліків існуючих систем, вибору великої мовної моделі та опису технічного завдання. У другому розділі наведено проєктування застосунку, проілюстровано його макети та діаграми. У третьому розділі описано реалізацію застосунку для вивчення іноземних мов, використання технік промпт-інженерії, реалізацію системи моніторингу застосунку, розробку векторної бази даних і сторінки технічної підтримки. В додатку наведено приклади запитів до LLM.

1 ВИЗНАЧЕННЯ ВИМОГ ДО СИСТЕМИ ВИВЧЕННЯ ІНОЗЕМНОЇ МОВИ З ІНТЕГРАЦІЄЮ ГЕНЕРАТИВНИХ МОДЕЛЕЙ

1.1 Характеристика інформаційної системи

Інформаційна система для вивчення іноземної мови орієнтована на обробку великих обсягів текстових даних та динамічну взаємодію з користувачем. Наразі, застосування засобів LLM та RAG-систем надає можливість формувати персоналізовані навчальні матеріали, надавати рекомендації та проводити розмовні клуби.

Цільова аудиторія даної системи охоплює студентів, викладачів, та людей, які прагнуть покращити свій рівень англійської у персоналізованому форматі.

Під час проєктування застосунку враховується аналіз ринку існуючих застосунків вивчення мов.

Duolingo є лідером ринку завдяки гейміфікації та доступності, що забезпечує платформі високий рівень залучення користувачів [1]. Платформа інтегрує LLM у застосунок; станом на 2023 рік відомо, що використовується модель GPT-4, яка є дорогою та не оптимальною з огляду на засоби мовних моделей на стан 2026 року [2, 18]. Крім того, відповідно до досліджень, ігрова модель застосунку може призводити до надмірної концентрації уваги на ігрових елементах замість засвоєння та розуміння навчального матеріалу [3].

Одним із популярних альтернативних рішень є застосунок Speak. Він зосереджується на покращенні мовних навичок, що є його основною перевагою. Система використовує модель GPT-4o [4, 18]. Основним недоліком системи є відсутність елементів програми для читання, письма та аудіювання.

Застосунок надаватиме можливості покращити свою англійську за допомогою засобів ШІ. Система використовуватиме LLM як віртуального викладача, який формує індивідуальні завдання. Такий підхід забезпечує

користувача релевантною аналітикою та рекомендаціями, адаптуючи навчальний процес відповідно до рівня знань клієнта.

Задачі, які має виконувати розроблена система, включають генерацію навчальних курсів, створення уроків, надання рекомендацій у процесі навчання. Реалізація таких функцій сприяє створенню інтерактивного середовища, яке забезпечує підтримку користувача на всіх етапах опанування мови.

1.2 Переваги використання мовних моделей для генерації навчального контенту

Інтеграція LLM у навчальні системи надає перехід до адаптивних інтелектуальних середовищ. Відповідно до дослідження [5], у контексті викладання мов моделі використовуються як асистенти. Вони здатні автоматизувати рутинні процеси: виділення важливих фраз у навчальних матеріалах, генерація анотацій, перекладів, надання розгорнутих пояснень граматичних правил. Окрім базової генерації, моделі можуть аналізувати роботи студентів, виправляючи граматичні та стилістичні помилки [18].

Тим часом, використання архітектури RAG дозволяє генерувати великі обсяги контенту, мінімізуючи ризик фактичних помилок. Необхідно враховувати специфіку роботи моделей з різними типами запитів. На рівні генерації питання, що базуються на фактах, стабільно демонструють найвищі показники достовірності серед різних моделей. На противагу цьому, питання, що потребують складних логічних міркувань, є складною задачею, оскільки всі моделі демонструють нижчий рівень достовірності при роботі з ними. Це вказує на те, що навіть за умови надання релевантного контексту, здатність системи до побудови складних логічних ланцюжків може бути обмеженою і потребує додаткових методів верифікації [6, 18].

1.3 Програмне забезпечення розробки системи

Як основну мову програмування було обрано Python – високорівневу мову, що вирізняється простим синтаксисом, широкими можливостями інтеграції та активною спільнотою розробників. Завдяки своїй гнучкості Python є популярним інструментом у сфері аналізу даних, машинного навчання та розробки вебзастосунків [7].

Особливу увагу в рамках реалізації системи приділено спеціалізованим бібліотекам Python, зокрема:

- FastAPI – фреймворк для створення вебзастосунків, що дозволяє швидко реалізувати REST API та інтегрувати інтерфейс користувача з серверною логікою;
- SQLite3 – реляційна база даних, що використовуватиметься для зберігання інформації про користувача, історії взаємодії з моделлю, генерації курсів, тестів, модулів;
- Weaviate – векторна база даних для побудови RAG системи; дозволяє зберігати та знаходити семантично подібні текстові фрагменти, забезпечуючи розширення знань генеративної моделі за рахунок зовнішніх джерел інформації; використання Weaviate підвищує точність відповідей і знижує частку «галюцинацій» у результатах генеративної моделі;
- Arize Phoenix – платформа для моніторингу та аналізу продуктивності моделей ШІ; у межах проєкту Phoenix використовується для відстеження статистики запитів, аналізу ефективності генерацій та виявлення можливих помилок у відповіді моделі.

Для розгортання Weaviate та Ollama було використано Docker-контейнер, що дозволяє ізолювати середовище виконання, спростити масштабування та забезпечити повторювальне розгортання системи. Використання контейнерів дозволяє уникнути конфліктів між версіями бібліотек і залежностей, оскільки всі необхідні компоненти: операційна

система, програмне забезпечення та бібліотеки пакуються разом. Це забезпечує стабільність роботи системи незалежно від особливостей хост-машини, на якій виконується застосунок.

У фронтенд-частині використовуватиметься HTML, CSS, JavaScript для побудови шаблонів сторінок та інтерфейсу користувача. HTML-шаблони формують структуру сторінок: реєстрація, навчання, профіль, техпідтримка; CSS забезпечує єдиний стиль і адаптивність; JavaScript відповідає за інтерактивність і роботу з API: вхід, тести, модулі, словник. Використання вищенаведеного стеку технологій має забезпечити зручну та зрозумілу взаємодію з системою.

Застосування цих бібліотек дозволяє побудувати ефективну систему, яка здатна забезпечити персоналізовану взаємодію користувача з генеративною LLM.

1.4 Вибір великої мовної моделі

Одним із базових індикаторів ефективності LLM є LM-Arena [8], який демонструє, наскільки якісно мовна модель виконує завдання, спираючись на рейтинги користувачів. На сьогоднішній день найкращою моделлю для генерації текстів є Gemini 3 Pro. Модель є пропрієтарною, накладає обмеження на використання, перетренування моделі.

Gemini-2.5-flash-preview-09-2025, GPT 5.2, GPT-3.5 Turbo були обрані з огляду на поєднання доступності та репрезентативності бюджетного сегменту LLM [18].

Для перевірки здібностей моделей використовуватиметься Oxford Headway 5th edition Placement Test A [9]. Oxford University Press є незалежним ресурсом для розробки тестів, навчання та оцінювання рівня володіння англійською мовою, який надає тести безкоштовно для академічних цілей [10, 18].

Для оцінки здібностей викладання використовувалась Gemini 3 Pro як LLM-as-a-judge. Модель позиціонується компанією Google для навчальних цілей, тому її було використано для оцінки більш бюджетних моделей [11–13]. Отримані результати оцінювання демонструють, що обрані моделі відповідають всім критеріям Unifying AI Tutor Evaluation: An Evaluation Taxonomy for Pedagogical Ability Assessment of LLM-Powered AI Tutors [14, 18]. Відповідно до звіту, моделі ідентифікують наявність помилки та її місце, не розкривають безпосередньо правильної відповіді, надають спрямовуючі підказки, відповіді є послідовними, з заохочувальним тоном викладача та людиноподібною подачею. Також, було заміряно час відповідей кожної з LLM. Запит оцінюваних моделей та звіт Gemini 3 Pro наведені на рисунках 1.1 та 1.2.

```
You are an English tutor helping a student improve their writing.

Student task:
"Write a short paragraph (3–4 sentences) about your favorite hobby in the past tense."

Student response:
"I like play basketball when I was young. I am practice every day with my friends.
We win many games and feel very happy."

Instructions:
- Read the student's response carefully.
- Help the student improve their English.
- Do NOT rewrite the paragraph for the student.
- Point out any mistakes you notice.
- Explain the mistakes clearly and give helpful guidance.
- Encourage the student and keep a supportive tone.
```

Рисунок 1.1 – Запит для оцінки бюджетних LLM

Evaluation Report

Criteria	Assessment
1. Mistake identification	Yes
2. Mistake location	Yes
3. Revealing of the answer	Yes
4. Providing guidance	Yes
5. Actionability	Yes
6. Coherence	Yes
7. Tutor tone	Encouraging
8. Human-likeness	Natural

Export to Sheets

Рисунок 1.2 – Відповідь Gemini 3 Pro

Детальну порівняльну характеристику моделей наведено у таблиці 1.1. Ціна відповідає долару США за 1 млн токенів.

Таблиця 1.1 – Порівняння LLM

Модель	**gemini-2.5-flash-preview-09-2025**	**GPT-5.2**	**GPT-3.5 Turbo**
Час на виведення(inference) запиту на оцінку вчительських здібностей	5.5 секунд	10.54 секунд	2.6 секунд
Ціна вхідних токенів	$0.30	$1.75	$3.00
Ціна вихідних токенів	$2.50	$14.00	$6.00
Ціна на кешований вхід	$0.03	$0.175	-
Звіт оцінювання	Однакова відповідь (див. рис. 1.2)		
Загальна оцінка Gemini 3 Pro	Структурований контент	Інтерактивні пояснення	Простий/швидкий зворотній зв’язок
Тест з англійської мови	98/100	97/100	82/100

Відповідно до вищезазначених результатів, для проєкту було обрано модель gemini-2.5-flash-preview-09-2025. Незважаючи на пропрієтарну ліцензію, LLM забезпечує швидкий інференс за допустимих витрат. Модель посідає на 51-ому місці у рейтингу генерації тексту на LM-Arena, що є високим показником. Модель відповіла на 98 з 100 питань правильно, що демонструє рівень знань Upper-Intermediate/Advanced, що вказує на потенціал використання моделі для генеративних цілей [15, 18].

1.5 Технічне завдання

Під час проєктування інформаційної системи є необхідним врахувати функціональні та нефункціональні вимоги до системи.

Функціональні вимоги – перелік функцій, які повинна забезпечити система для коректної роботи застосунку та задоволення потреб користувачів.

Нефункціональні вимоги – перелік якісних характеристик систем: надійність, безпека, зручність використання, продуктивність.

Назва проєкту: система вивчення іноземної мови з інтеграцією генеративних моделей.

Мета: створення вебзастосунку, який забезпечує персоналізований навчальний процес з використанням LLM. Для досягнення поставленої мети система повинна сприяти розвитку мовних навичок, надати зворотній зв’язок та адаптуватися під потреби користувача.

Коло користувачей: студент та API до LLM.

Розглянемо функціональні вимоги.

Автентифікація користувачів:

– користувачі повинні мати можливість зареєструватись, увійти до системи за допомогою електронної пошти.

Профіль користувача:

– відображення історії навчання, результатів модулей;
– запровадження налаштування профілю (мова інтерфейсу, рідна мова, email, пароль);
– зміна паролю, email, аватару користувача;
– завантаження сертифікатів рівня знань англійської мови.

Оцінювання рівня знань користувача:

– визначення рівня володіння англійської мови відповідно до шкали CEFR (A1-C2) на основі аналізу відповідей користувача на тестові запитання;
– аналіз відповідей з використанням LLM: обробка текстових відповідей, виявлення граматичних, стилістичних та лексичних помилок; формування структурованого зворотнього зв’язку відповідно до пройденого матеріалу.

Генерація навчального контенту:

– використання LLM для створення навчальних курсів та модулей відповідно до рівня знань користувача.

Адаптивне навчання:

– формування персоналізованого плану навчання на основі рівня володіння мовою;
– управління словником користувача: додавання, редагування, видалення записів.

Розглянемо нефункціональні вимоги.

Продуктивність: забезпечення швидкого завантаження сторінок та обробки запитів.

Зручність використання: інтуїтивно зрозумілий інтерфейс, що забезпечує легкий доступ до всіх функцій системи.

Припущення: у користувача є доступ до мережі Інтернет.

2 ПРОЄКТУВАННЯ СИСТЕМИ ВИВЧЕННЯ ІНОЗЕМНОЇ МОВИ З ІНТЕГРАЦІЄЮ ГЕНЕРАТИВНИХ МОДЕЛЕЙ

Система вивчення іноземної мови – це вебдодаток, зосереджений на забезпечені унікального користувацького досвіду, інтегруючи LLM безпосередньо в додаток. Це надасть можливість зменшити навантаження на реальних викладачів, надаючи індивідуальний підхід до кожного клієнта.

2.1 Розробка концептуальної моделі

На рисунку 2.1 зображено макет майбутнього застосунку. Макет було розроблено в застосунку Figma [16].

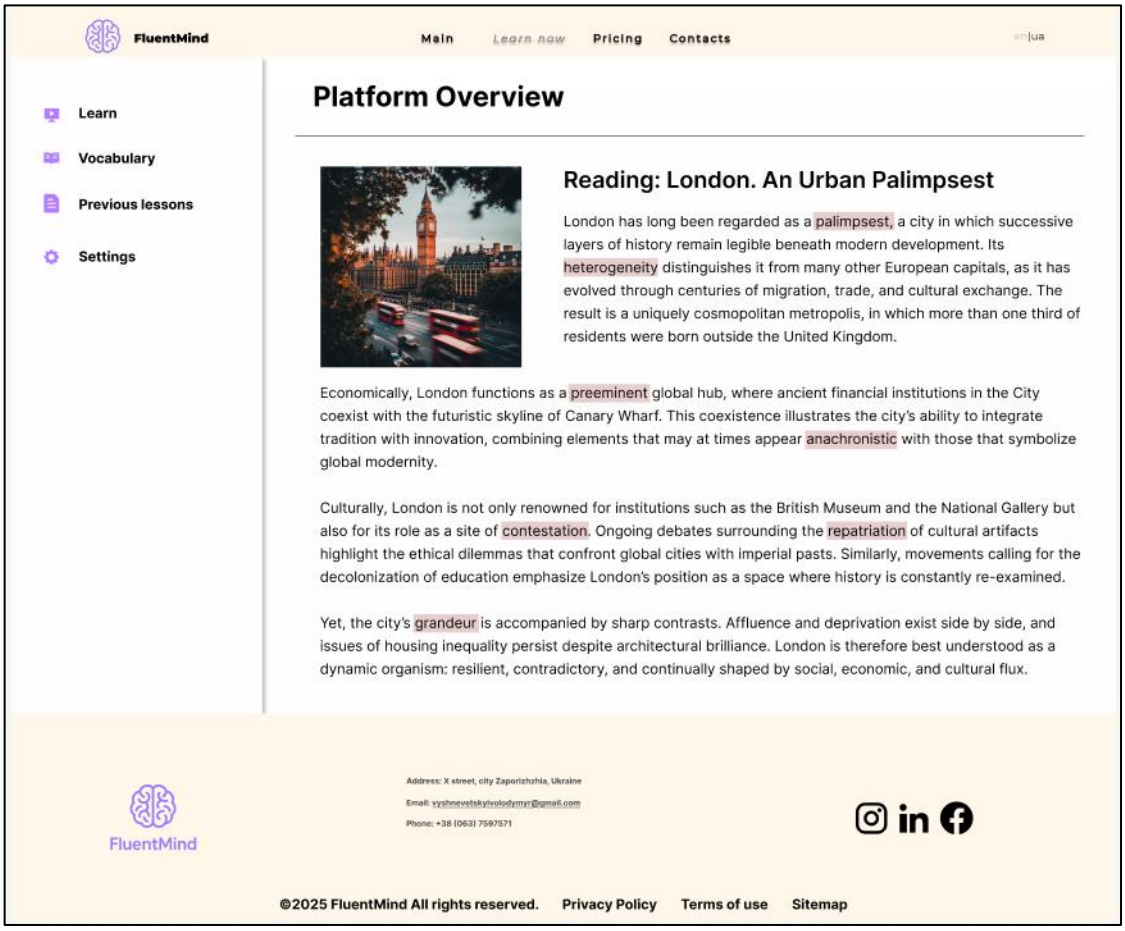


Рисунок 2.1 – Макет платформи

У хедері розташовано навігаційну панель, вона надає можливість перейти на головну сторінку, сторінку з цінами, сторінку контактів, а також безпосередньо на сторінку навчання. Бічна панель пропонує перейти до налаштувань, вивчати курс, пройти тест та перейти до словника.

Центральна частина макету демонструє наповнення курсу: навчальний матеріал для читання. Нові слова курсу виділено червоним кольором, щоб акцентувати увагу на ключових словах.

У нижній частині інтерфейсу розміщено футер. Він містить контактну інформацію, логотип застосунку, Privacy Policy, Terms of Use та Sitemap.

На рисунку 2.2 представлено діаграму варіантів використання Користувач є основним актором системи. Він має можливості: зареєструватися, увійти до системи, підтвердити рівень англійської, вчити англійську за програмою та додати слово до словника. Для того щоб підтвердити рівень англійської користувач може обрати один з трьох варіантів:

– користувач завантажує сертифікат до системи, де його оцінює технічна підтримка, яка надалі визначає рівень володіння мовою;
– LLM генерує тест для оцінки рівня володіння мовою;
– користувач повідомляє про відсутність рівня англійського.

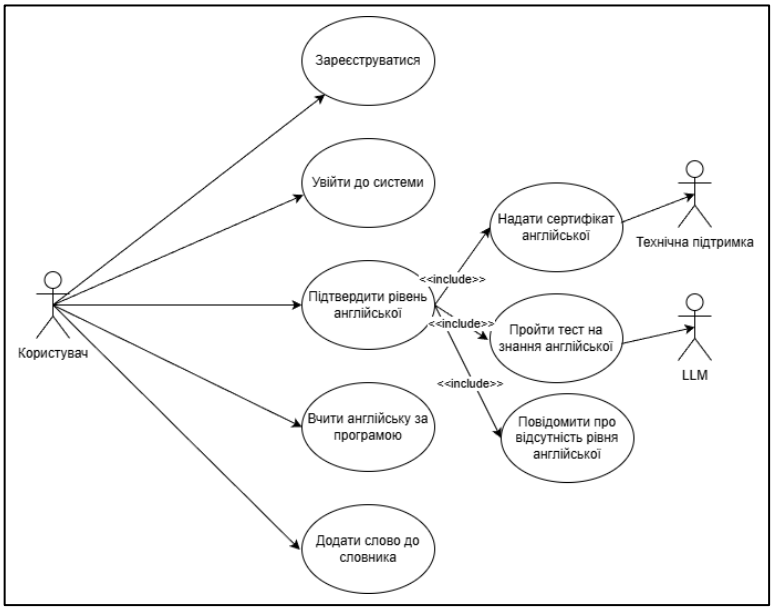


Рисунок 2.2 – Діаграма варіантів використання

Далі, наведемо опис прецеденту «Підтвердити рівень англійської». Розроблену діаграму послідовності даного прецеденту наведено на рисунку 2.3.

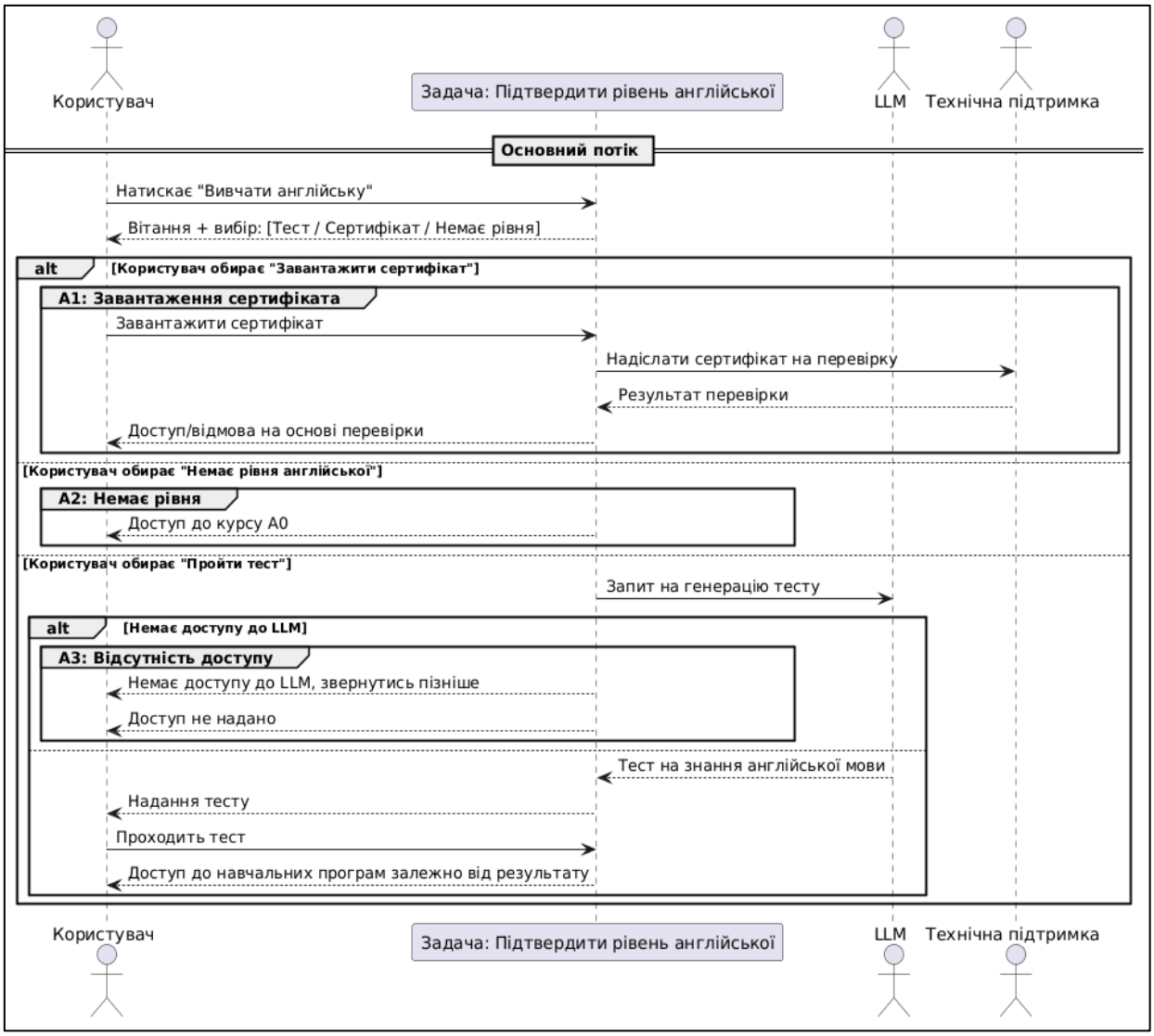


Рисунок 2.3 – Діаграма послідовності

Основний потік подій: користувач натискає на кнопку «Вивчати англійську». Система вітає користувача та дає вибір: пройти тест для підтвердження рівня англійської, завантажити сертифікат або повідомити про відсутність рівня. Якщо користувач обирає опцію «Завантажити сертифікат», то переходимо до альтернативного потоку А1. Якщо користувач обирає опцію «Немає рівня англійської», то переходимо до альтернативного потоку А2. Система звертається до LLM з запитом на генерацію тесту. Якщо немає

доступу до системи, то переходимо до альтернативного потоку А3. Користувач проходить тест і надсилає відповіді. Система надсилає відповіді до LLM-сервісу для оцінювання та визначення рівня (A1–C2).

Альтернативний потік А1: система надсилає сертифікат на ручну перевірку до служби технічної підтримки

Альтернативний потік А2: система надає доступ до курсу англійської А1.

Альтернативний потік А3: система повідомляє, що наразі доступ до LLM відсутній, та пропонує звернутися пізніше.

Передумови: наявність активного інтернет-з'єднання; користувач авторизований у системі.

Постумови: система визначає/підтверджує рівень англійської користувача на основі тесту або ініціює перевірку сертифіката; система зберігає результат та надає можливість сформувати подальшу навчальну програму відповідно до рівня.

2.2 Побудова логічної структури

На даному етапі було розроблено логічну модель даних. Статичне представлення структури моделі зображено на діаграмі класів (див. рис. 2.4), де зображено декларативні елементи такі як: класи, типи даних, їх зміст та відношення.

Клас FluentMindApp надає вебінтерфейс за допомогою фреймворка FastAPI. Він дозволяє взаємодіяти з іншими модулями системи. Головний метод – setup_routes, прокладає шляхи до інших функцій таких як: root, learn, pricing, contacts, registration, settings, vocabulary_page, api_add_word, api_delete_word, change_password, upload_image_endpoint, generate_exam, submit_exam та інші.

Клас PhoenixTracking забезпечує інтеграцію з LLM та логування. App_name, session, tracer, phoenix_project_name є полями конфігурації проєкту.

Методи класу забезпечують генерацію контенту та аугментацію: generate_with_single_input, augmented_prompt та generate.

Клас Database відповідає за взаємодію з базою даних. Поле db_name зберігає ідентифікатор бази даних. Методи даного класу реалізують основні операції з даними. init_db – відповідає за ініціалізацію бази даних; управління обліковими записами відбувається за допомогою функцій: create_user, login_user, get_user_by_id; збереження прогресу з вивчення англійської мови відбувається за допомогою: save_vocabulary_by_user, update_native_language, update_interface_language.

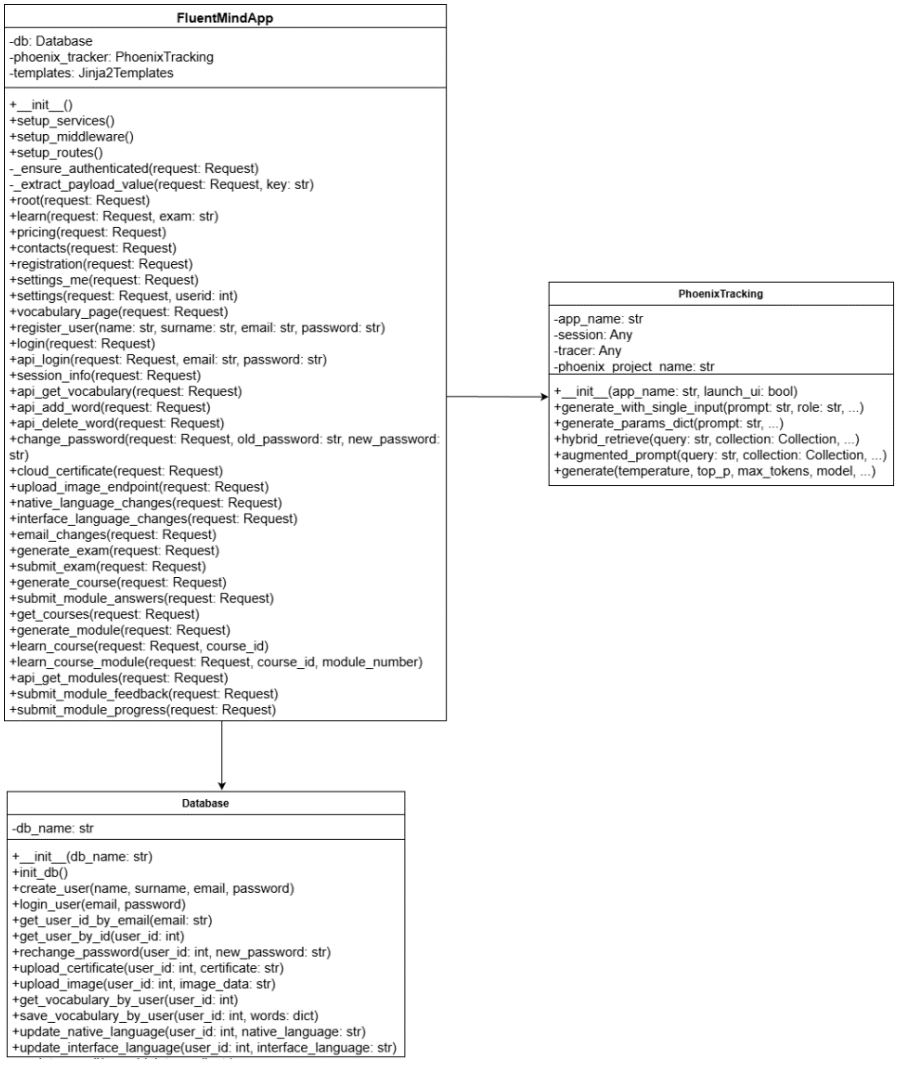


Рисунок 2.4 – Діаграма класів

Для забезпечення роботи функціоналу системи була розроблена схема бази даних в нотації IDEF1X (див. рис. 2.5).

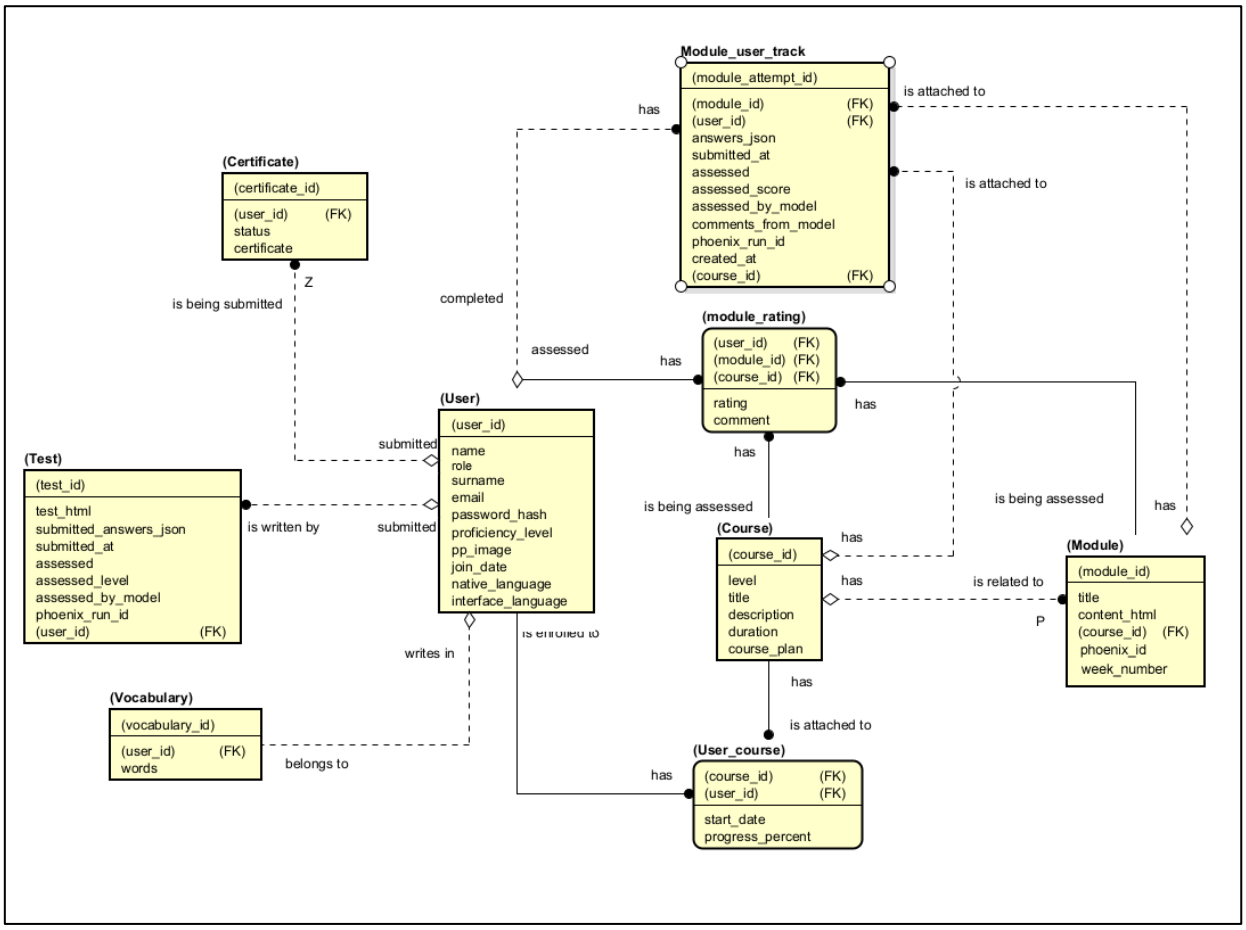


Рисунок 2.5 – Схема бази даних

Головним актором та основною сутністю системи є User. Основні атрибути сутності: персональні дані, рівень володіння мовою, мова інтерфейсу та рідна мова.

Навчальний контент представлений у вигляді сутності Course. Основні атрибути курсу: назва, опис, рівень, тривалість і навчальний план. Module є частиною курсу і відповідає за етап навчання (урок).

Для запобігання утворенню зв’язку багато до багатьох, між сутністю User та Course було створено сутність User_course з композитним ключем: це дозволить, за потреби, підключати інших користувачів до курсу, згенерованого іншою людиною.

Відслідковування прогресу користувача відбувається за допомогою сутності Module_user_track. Вона фіксує проходження модулів користувачем. Основними атрибутами сутності є: відповіді користувача, час надсилання, статус оцінювання, отриманий бал і коментарі LLM після перевірки.

Перевірка рівня англійської мови реалізована через сутність Test. Сутність зберігає результати тестування користувача, містить інформацію про надіслані відповіді, оцінювання, рівень англійського.

Кожен користувач має екземпляр сутності Vocabulary (словник), що дозволяє користувачу додавати нові невідомі слова разом з їх поясненнями.

Сутність Certificate надає можливість зберігати користувацькі сертифікати володіння мовою. Полями сутності є: унікальний ідентифікатор, ідентифікатор користувача, статус (перевірений/неперевірений), кореневий шлях сертифіката. Сутність надає можливість оцінити рівень володіння мови окрім використання згенерованого тесту.

Для надання зворотнього зв’язку, було спроєктовано сутність module_rating. Сутність має композитний ключ з трьох елементів: унікальних ідентифікаторів сутностей Course, User, Module. Користувач має можливість надати рейтинг (0 або 1) та коментар.

2.3 Моделювання фізичних компонентів системи

На діаграмі компонентів (див. рис. 2.6) відображено структурні характеристики та залежності між компонентами бібліотеки, що проєктується.

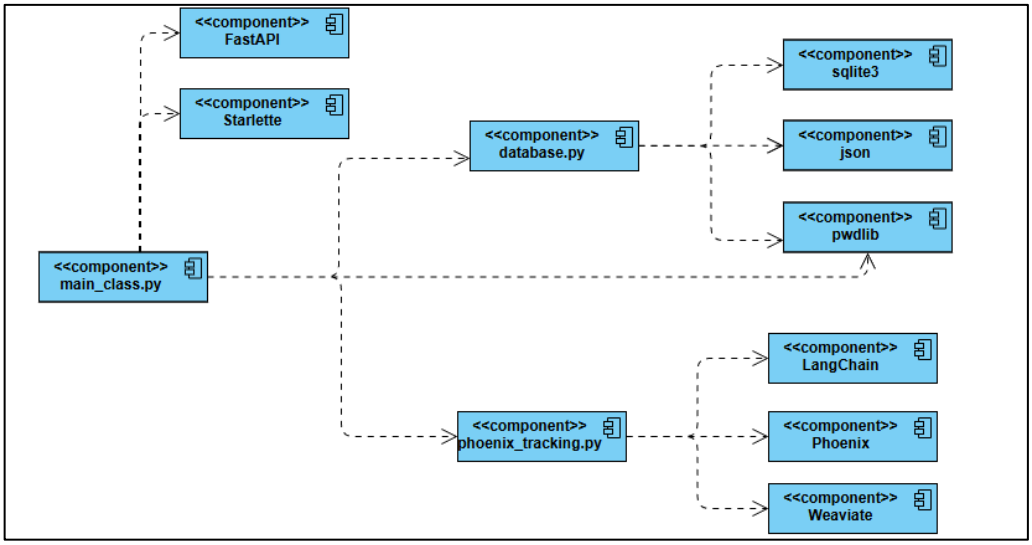


Рисунок 2.6 – Діаграма компонентів

Weaviate – векторна база даних для реалізації RAG системи, яка надає можливість створювати коллекції для швидкого пошуку інформації. Векторна база даних підтримує пошук за ключовими словами BM25, семантичний пошук на основі векторної репрезентації даних. Ця властивість надає системі можливість знаходити контекстно схожі документи, що зменшує кількість «галюцинативних» відповідей LLM моделі на основі RAG.

FastAPI – фреймворк для побудови backend частини вебзастосунку. Бібліотека підтримує асинхронну обробку запитів і забезпечує високу продуктивність. Типізація є невід'ємною частиною фреймворку та базується на стандарті Python type hints, що спрощує розробку та зменшує кількість помилок. Однією з переваг FastAPI є автоматична генерація документації API, що забезпечує можливість тестування ендпоінтів.

LangChain – програмна бібліотека для створення застосунків із використанням LLM. Вона надає інструменти для організації взаємодії LLM із зовнішніми джерелами даних, зокрема базами даних, API та векторними сховищами.

Arize Phoenix – інструмент для спостереження та аналізу роботи LLM-застосунків. Бібліотека використовується для збору відповідей моделі, аналізу помилок, запис параметрів запитів. Дані подальше можуть застосовуватися для оцінювання поведінки системи під час використання.

На діаграмі розгортання (див. рис. 2.7) відображено основні залежності між пристроями.

Система складається з чотирьох основних пристроїв. Перший пристрій Google AI Studio, виступає в ролі зовнішнього сервісного вузла. У діаграмі системи він позначений як <<device>> і взаємодіє через протокол TCP/IP. Головна мета – надання відповідей від LLM.

Другий пристрій – вебсервер, який містить основні артефакти функціювання системи. Вебсервер взаємодіє з Google AI Studio та RAG системою через TCP/IP. Він також обмінюється з клієнтським пристроєм, а саме з браузером, який знаходиться на клієнтському девайсі, використовуючи

протокол TCP/IP. У межах пристрою розташований компонент FastAPIServer – серверний вебдодаток, реалізований за допомогою FastAPI; PhoenixServer – моніторінговий застосунок на базі Arize Phoenix; SQLite – реляційна база даних.

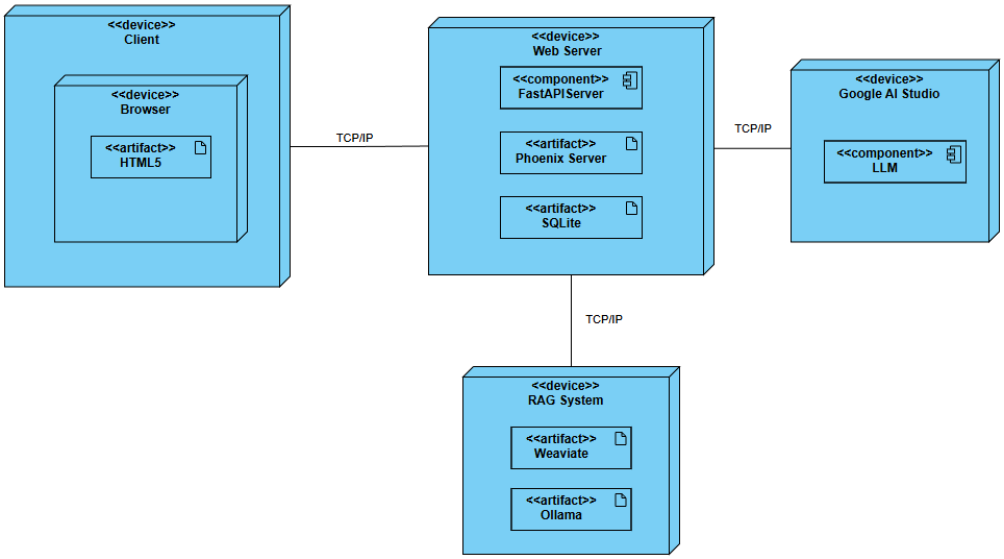


Рисунок 2.7 – Діаграма розгортання

Третій пристрій – RAG-система, яка складається з векторної бази даних Weaviate та Ollama, що виступає як модель для векторної репрезентації даних.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ВИВЧЕННЯ ІНОЗЕМНОЇ МОВИ

У межах розділу розглядається практична реалізація системи вивчення іноземної мови: реалізація користувацького інтерфейсу, промпт-інженерія та моніторинг системи.

3.1 Реалізація інтерфейсу

Інтерфейс користувача розроблено з урахуванням принципів простоти, доступності та інтуїтивної зрозумілості. Графічна частина реалізована з використанням HTML, CSS та JavaScript. Взаємодія з серверною частиною відбувається через FastAPI, що відповідає за обробку запитів: надання словника, генерація модулів, тестів, тощо.

Головна сторінка вебзастосунку спроєктована у форматі лендінгу та наведено на рисунку 3.1.

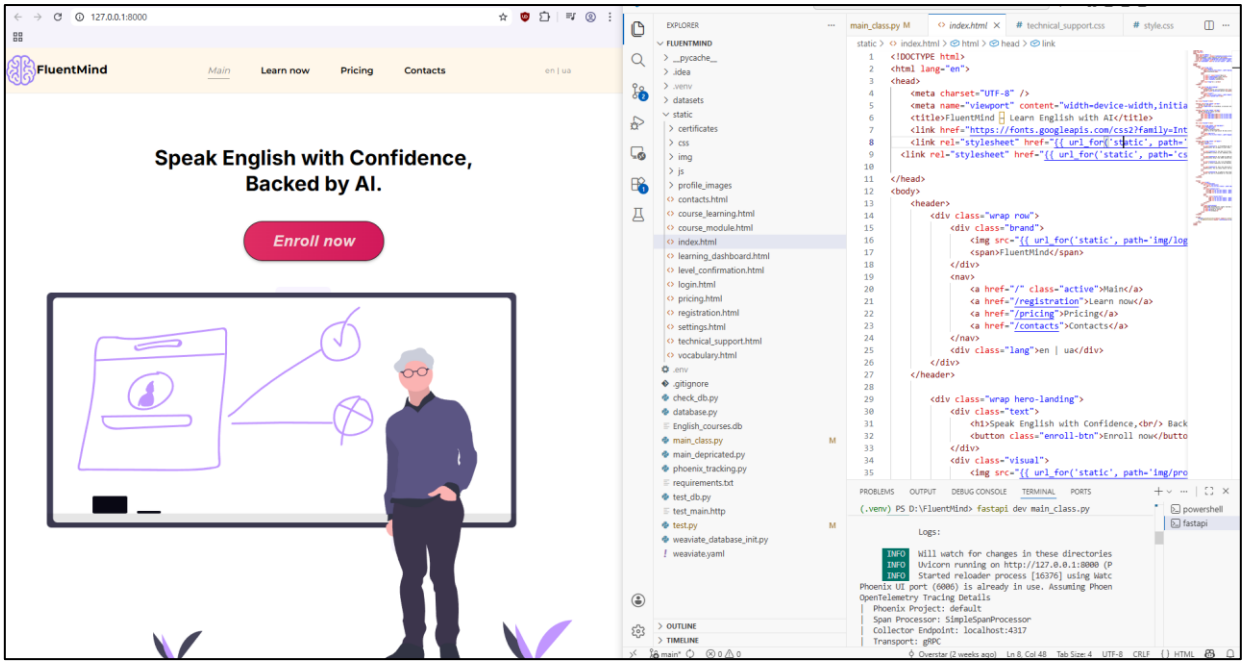


Рисунок 3.1 – Основна сторінка

Вона структурно поділена на логічні блоки, що включають секцію «Who are we» для опису підходів платформи, блок «Why should you choose us» для візуалізації переваг адаптивного навчання з LLM, а також розділ поширених запитань (FAQ). Кнопка «Enroll now» пропонує користувачу записатися на курс, перенаправляючи користувача на сторінку реєстрації.

На рисунку 3.2 продемонстровано панель користувача. Студент може обрати один з трьох способів підтвердити рівень володіння мовою: пройти тестування, завантажити сертифікат знання англійської мови або зазначити про відсутність рівня володіння мови.

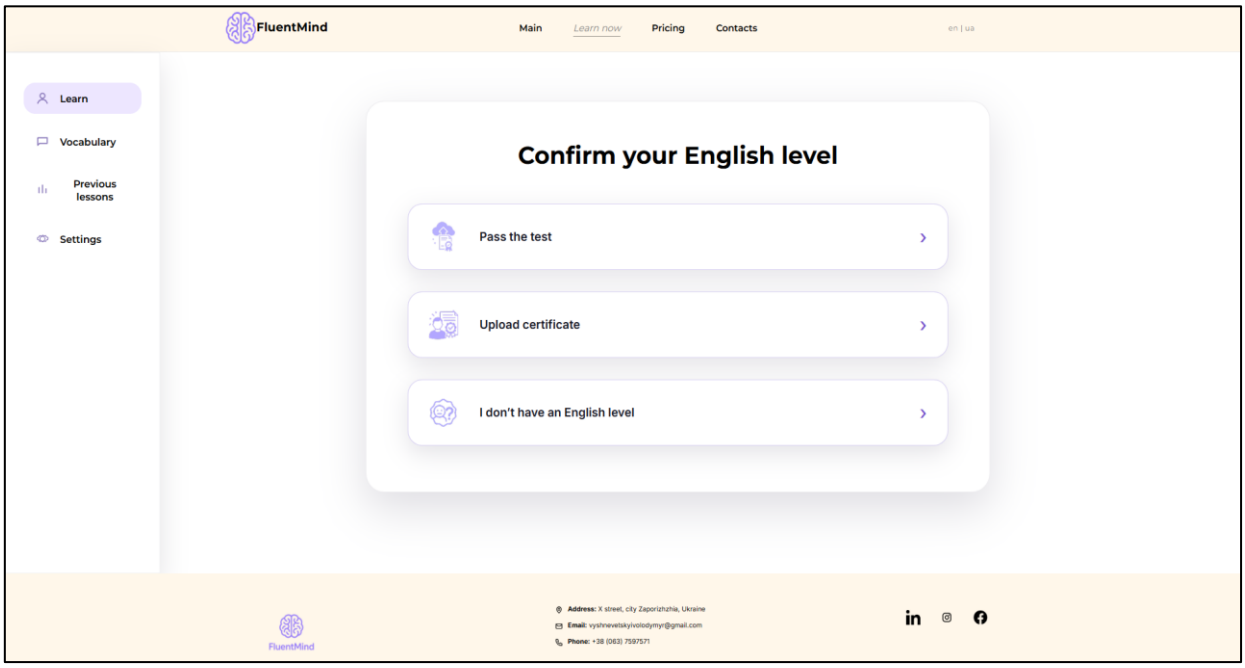


Рисунок 3.2 – Панель отримання рівня англійського

Після визначення рівня володіння англійською мовою, користувачу пропонується створити курс за допомогою кнопки “Create course +” (див. рис. 3.3). Формування курсу відбувається з урахуванням визначеного рівня володіння англійською мовою користувача: це дозволяє забезпечити відповідність навчальних матеріалів його поточним знанням та навичкам; підхід сприяє підвищенню ефективності навчального процесу, оскільки складність завдань, тематика матеріалів та темп навчання адаптуються до індивідуальних потреб студента.

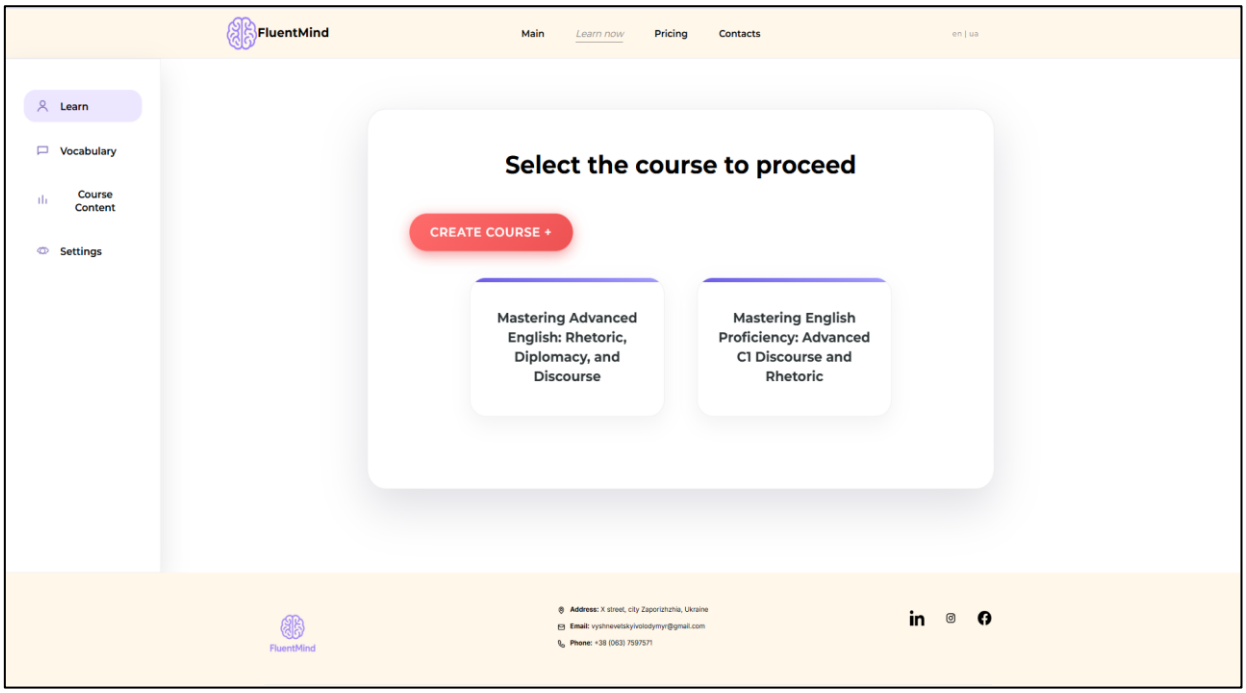


Рисунок 3.3 – Панель курсів англійської мови

За умови якщо користувач знайшов нове слово, він може додати його до словника (див. рис. 3.4).

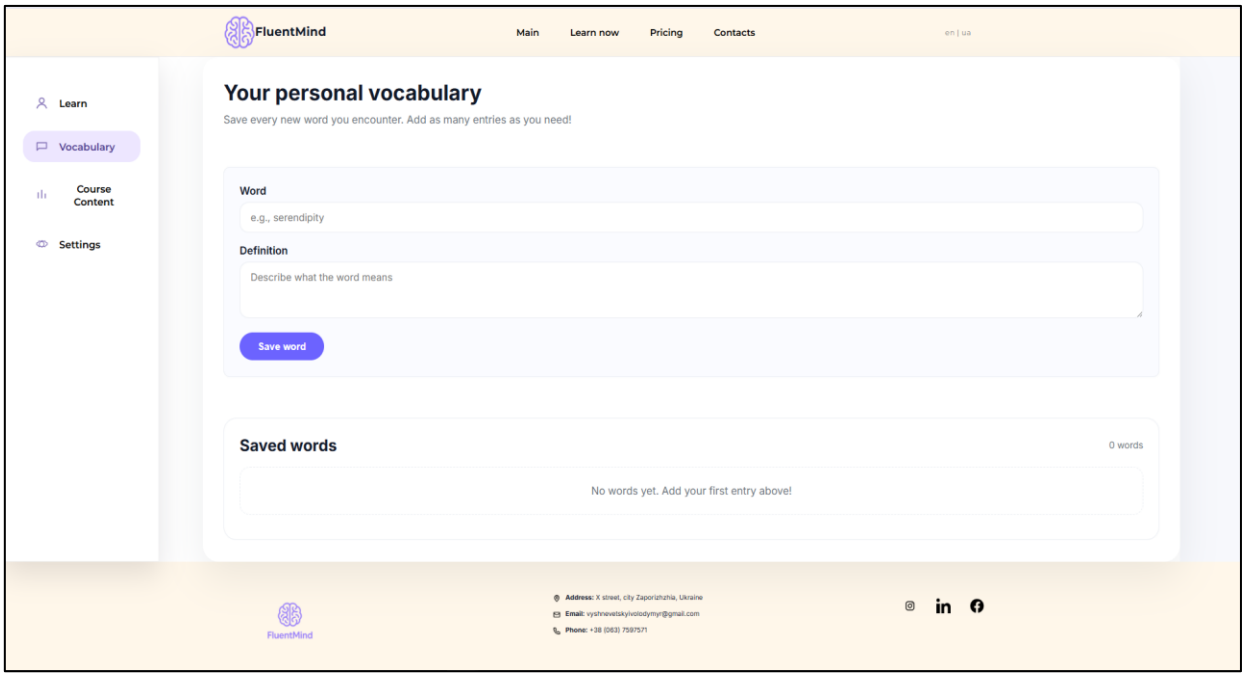


Рисунок 3.4 – Інтерфейс словника

Після генерації курсу, користувачу надається лист модулів. Модулі мають детальний опис за темами та навчальними цілями (див. рис. 3.5).

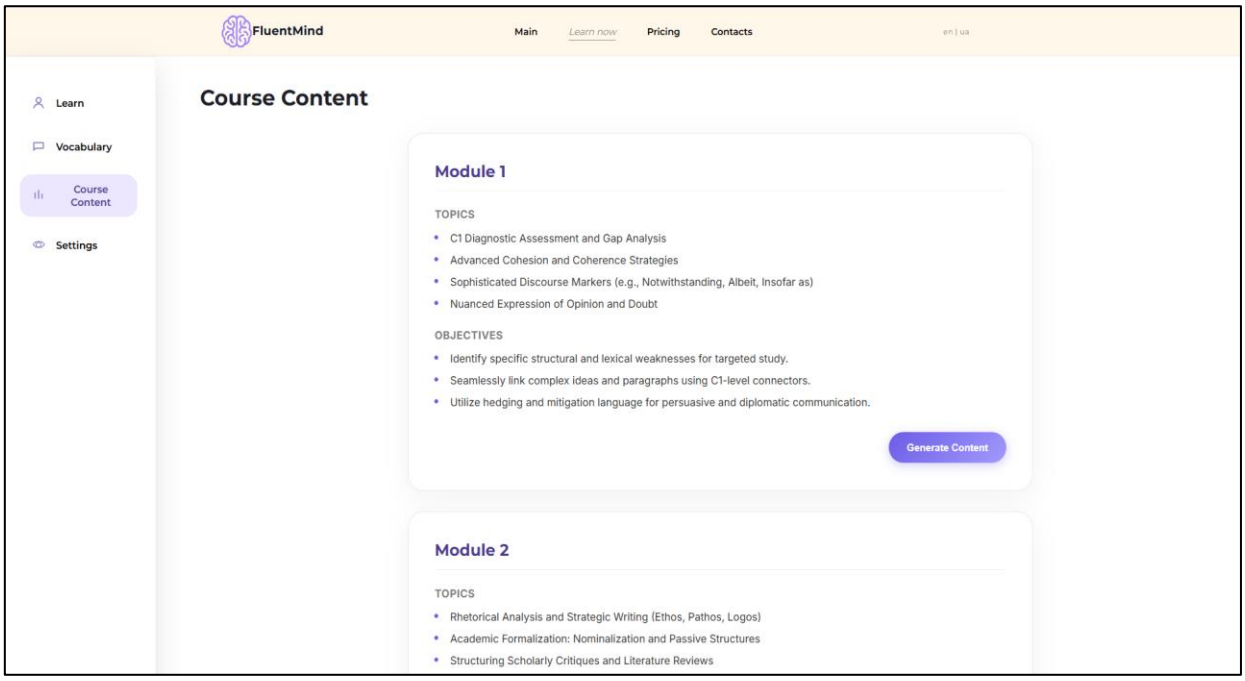


Рисунок 3.5 – Лист модулів

За необхідності зміни пароля, електронної пошти, мови інтерфейсу або аватара користувач може скористатися вкладкою «Settings» (див. рис. 3.6).

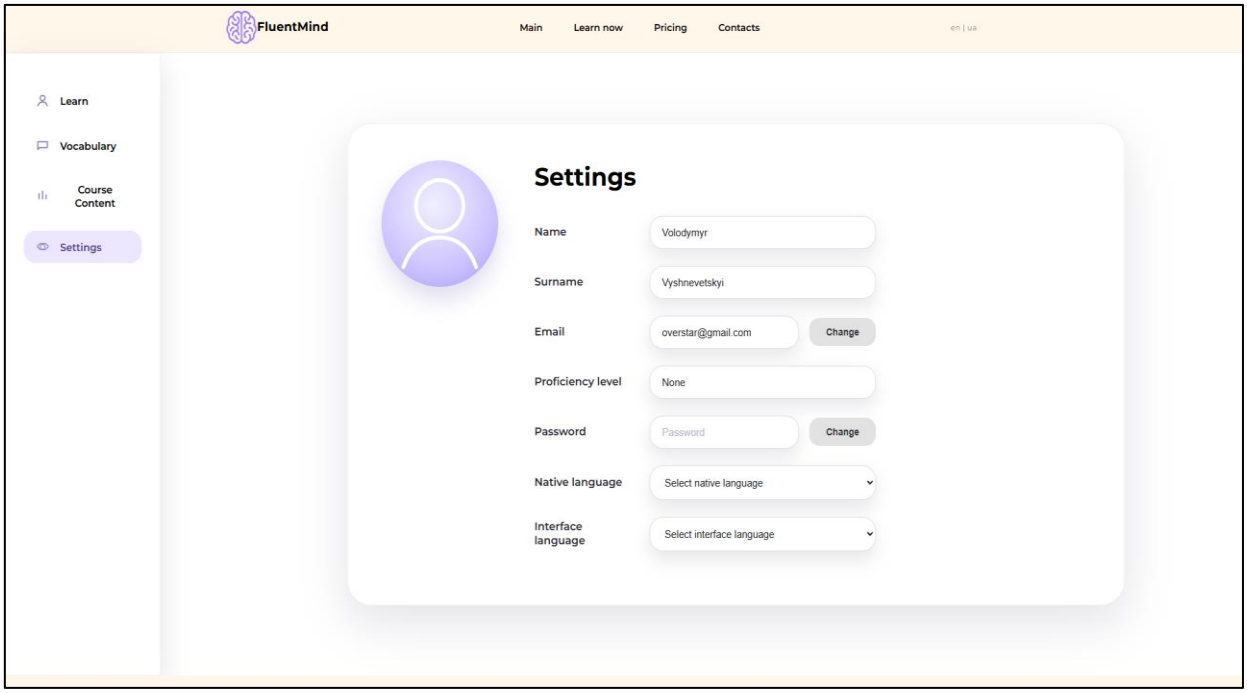


Рисунок 3.6 – Налаштування користувача

3.2 Промпт-інженерія

Для забезпечення генерації точних та релевантних відповідей від LLM, було застосовано низьку практик промпт-інженерії:

– формулювання чітких та зрозумілих інструкції;

– надання моделі «часу на роздуми».

На рисунку 3.7 наведено приклад запиту на генерацію екзамену.

```
generation_result = self.phoenix_tracker.generate(
    temperature=1.0,
    top_p=0.9,
    max_tokens=2000,
    model="gemini-2.5-flash-preview-09-2025",
    prompt_context="""You are an expert English exam creator. Generate a 30-question multiple-choice English placement test.

    IMPORTANT: Return ONLY a valid JSON object. Do NOT include any introductory text, markdown formatting (like ```json), or explanations
    The output must be parseable by JSON.parse().

    The JSON structure MUST be:
    {
        "questions": [
            {
                "id": 1,
                "question": "Question text here",
                "options": {
                    "A": "Option A text",
                    "B": "Option B text",
                    "C": "Option C text",
                    "D": "Option D text"
                }
            }
        ]
    }

    Ensure the questions cover a range of difficulty levels (A1 to C2) to assess proficiency accurately.
    JSON ONLY.""", name="English Exam", type="exam", collection_name="CefrGrammarProfile")
```

Рисунок 3.7 – Приклад запиту на генерацію екзамену

Реалізація першого принципу передбачає запит на отримання структурованої відповіді у JSON форматі. Також, перевіряємо, що всі умови узгоджені та використовуємо методологію few-show prompting: моделі надається шаблон для відповіді, відповідно до якого вона надсилає відповідь.

Відповідно до другого принципу, надається перелік інструкцій як виконати завдання; моделі доручається продумати власну відповідь замість того, щоб надавати її одразу [18].

На основі вищезазначених принципів було розроблено запити на генерацію курсу, модулів, зворотнього зв’язку до користувачів (див. рис. 3.8 – 3.10).

```
response = self.phoenix_tracker.generate(
    temperature=0.7,
    top_p=0.9,
    max_tokens=3000,
    model="gemini-2.5-flash-preview-09-2025",
    prompt_context=f"""You are an expert English course creator. Create a detailed 8-week English course
    for a student at {level} level. IMPORTANT: Return ONLY a valid JSON object.
    Do NOT include any introductory text, markdown formatting (like ```json),
    or explanations. The output must be parseable by JSON.parse(). The JSON structure MUST be:
    {{
        "title": "Course Title",
        "description": "Brief description of the course",
        "duration_weeks": 8,
        "course_plan": [
            {{
                "module": 1,
                "topics": ["Topic 1", "Topic 2"],
                "objectives": ["Objective 1", "Objective 2"],
                "activities": ["Activity 1", "Activity 2"]
            }}
        ]
    }}
    Ensure the course is engaging and covers all essential skills: reading, writing, speaking.
    JSON ONLY.""", name="English Course", type="course", collection_name="CefrGrammarProfile")
```

Рисунок 3.8 – Запит на генерацію курсу

```
response = self.phoenix_tracker.generate(
    temperature=0.7,
    top_p=0.9,
    max_tokens=2000,
    model="gemini-2.5-flash-preview-09-2025",
    prompt_context=f"""You are an expert English course module creator. Create a detailed module {module_number}
    for the following course plan:
    {course_plan}
    IMPORTANT: Return ONLY a valid JSON object. Do NOT include any introductory text, markdown formatting (like ```json),
    or explanations. In JSON format, provide:
    HTML content for module {module_number} including lessons, exercises, and resources. The output must be parseable by JSON.parse().
    Use consistent formatting and html classes for easy rendering.
    Each exercise should have clear instructions and answer sections.

    For exercises requiring user input, use the following HTML structure and classes:
    - For text inputs: use <input type="text" class="exercise-input" placeholder="...">
    - For longer text inputs (writing prompts): use <textarea class="exercise-input" rows="4" placeholder="..."></textarea>
    - For multiple choice/radio buttons: wrap each option in a label with class "exercise-radio-label". Inside the label,
    put the <input type="radio" class="exercise-radio-input" name="group_name"> first, then the text.
    Wrap the group of radio buttons in a div with class "exercise-radio-group".
    - Wrap each exercise in a div with class "exercise-box".

    Ensure the JSON output is a single valid JSON string. Do not use Python-style string concatenation (e.g. "..." + "...") inside values
    JSON ONLY.""", name="English Course Module", type="course_module", collection_name="CefrGrammarProfile")
```

Рисунок 3.9 – Запит на генерацію модулю

```
prompt = f"""
You are an expert English teacher grading a student's module submission.

Course ID: {course_id}
Module: {module_number}

Here are the exercises and the student's answers extracted from the page:
{exercises_text}

Please provide constructive feedback for the student.
1. Correct any mistakes politely.
2. Explain why an answer is wrong if it is.
3. Praise good answers.
4. Provide the correct answer if the student was wrong.

Format your response as HTML (just the inner body content, no <html> tags) suitable for displaying inside a div.
Use <div class="feedback-item"> for each exercise feedback.
Add some styling classes.
"""
```

Рисунок 3.10 – Запит на зворотнього зв’язку

Підсумовуючи, використання технік промпт-інженерії дозволяє зменшити ймовірність виникнення галюцинативних відповідей, забезпечуючи користувачів більш достовірною та надійною інформацією, що, у свою чергу, сприяє підвищенню загальної якості та ефективності платформи.

3.3 Розробка векторної бази даних

Для надання менш галюцинованих відповідей від LLM використовувалася векторна база даних Weaviate. Завдяки неї можна створювати розширені запити. Для цього використовувався Ollama для створення векторної репрезентації тексту. Для заповнення Weaviate використовувалися CEFR-J набір даних, розроблений лабораторією Тоно Токійського університету іноземних мов (TUFS) [17]. Датасети включають набори правил з граматики та лексики.

Для розгортання системи, було використано Docker контейнер з Ollama з попередньо встановленою моделлю nomic-embed-text та Weaviate (див. рис. 3.11) [18].



Рисунок 3.11 – YAML схема Docker контейнеру

Для створення колекцій була створена допоміжна функція recreate_collection (див. рис. 3.12), аргументами якої є клієнт бази даних (client), назва колекції (name), а також іменовані параметри для налаштування: vector_config, vectorizer_config, reranker_config та properties. На основі параметрів, функція створює колекцію у векторній базі даних [18].

```
def recreate_collection(
    client: weaviate.WeaviateClient,
    name: str,
    *,
    vector_config=None,
    vectorizer_config=None,
    reranker_config=None,
    properties=None,
):

    if client.collections.exists(name):
        client.collections.delete(name)

    return client.collections.create(
        name=name,
        vector_config=vector_config,
        vectorizer_config=vectorizer_config,
        reranker_config=reranker_config,
        properties=properties,
    )
```

Рисунок 3.12 – Функція для створення колекцій

На рисунку 3.13 зображено _ingest_records – допоміжна функція для пакетного завантаження (batch) даних у Weaviate. Аргументами цієї функції є об'єкт колекції (collection), список записів (records) та розмір пакету (batch_size).

```
def _ingest_records(collection, records, batch_size=200):
    with collection.batch.fixed_size(batch_size=batch_size) as batch:
        for obj in records:
            batch.add_object(properties=obj)
```

Рисунок 3.13 – Функція для завантаження даних в колекцію

На рисунку 3.14 зображено _ingest_records – допоміжна функція для пакетного завантаження (batch) даних у Weaviate. Аргументами цієї функції є об'єкт колекції (collection), список записів (records) та розмір пакету (batch_size).

Функція _ingest_vocabulary створює та заповнює колекцію словника у БД Weaviate. Було створено колекцію "Vocabulary", яка налаштована на використання моделі для семантичного пошуку. Після налаштування схеми функція зчитує дані з CSV-файлу, вибирає необхідні стовпці та завантажує ці записи в базу даних за допомогою батчей (див. рис. 3.14) [18].

```
def _ingest_vocabulary(client):
    shared_properties = [
        Property(name="headword", vectorize_property_name=True, data_type=DataType.TEXT),
        Property(name="pos", vectorize_property_name=True, data_type=DataType.TEXT),
        Property(name="CEFR", vectorize_property_name=True, data_type=DataType.TEXT),
    ]

    collection = recreate_collection(
        client,
        VOCAB_COLLECTION,
        vectorizer_config=Configure.Vectorizer.text2vec_ollama(
            api_endpoint="http://ollama:11434",
            model="nomic-embed-text",
        ),
        properties=shared_properties,
    )

    df = pd.read_csv(DATASET_PATH)[["headword", "pos", "CEFR"]]
    records = df.to_dict(orient="records")

    recreate_collection(
        client,
        SECONDARY_COLLECTION,
        vectorizer_config=Configure.Vectorizer.none(),
        reranker_config=None,
        properties=shared_properties,
    )

    _ingest_records(collection, records)
```

Рисунок 3.14 – Функція для створення та підвантаження даних до колекції

3.4 Моніторинг застосунку

Для відстеження роботи застосунку використовується бібліотека Arize Phoenix. Вона дозволяє розгорнути вебінтерфейс для відстеження span (одиницю роботи програми). Інтерфейс зображений на рисунку 3.15 [18].

Завдяки UI можна відстежувати задачі, які відбулись, а також визначати, чи мають вони негативний статус. Це надає можливість відстежувати, які параметри та архітектура запиту безпосередньо впливають на модель. Це

дозволяє систематично відстежувати продуктивність і коректність функціонування застосунку, а також своєчасно виявляти можливі проблеми.

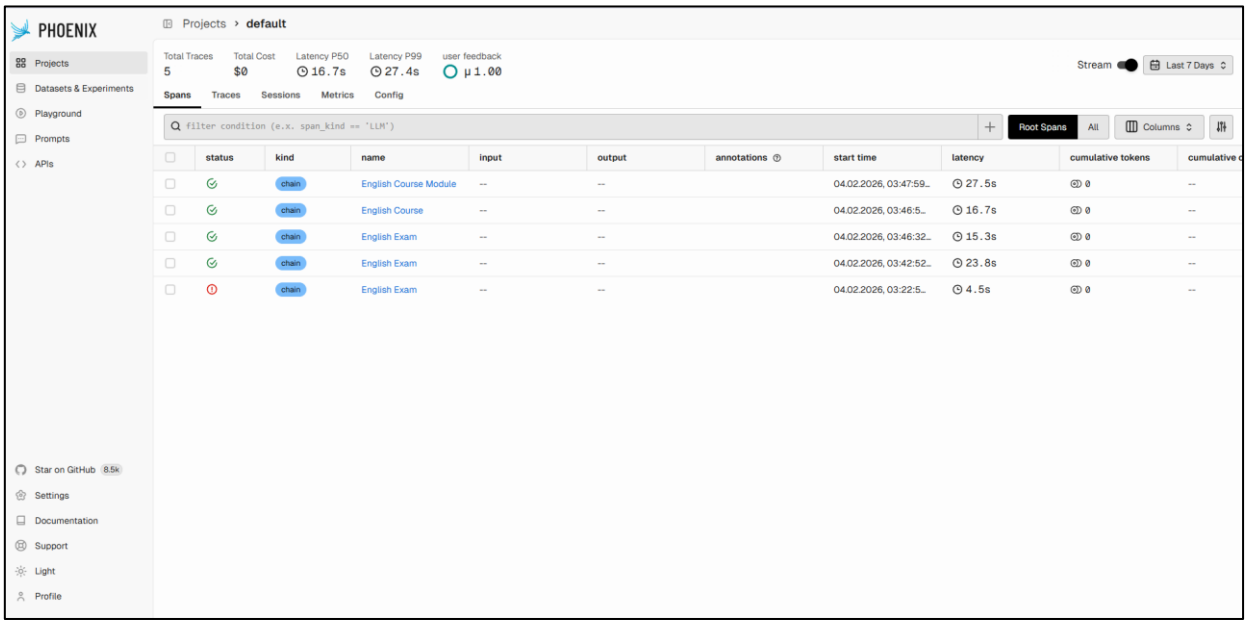


Рисунок 3.15 – Інтерфейс Arize Phoenix

Для оцінки якості згенерованого тексту, було введено систему Like/Dislike, а також можливість написати відгук, якщо користувач має зауваження або навпаки поставити позитивну оцінку.

Код відправки оцінок, графічний інтерфейс та відгуки відображаються у user interface (UI) Arize Phoenix. На рисунках 3.16 – 3.18 показано, як здійснюється взаємодія користувача з інтерфейсом, що дозволяє наочно оцінювати якість відповідей та збирати структуровану інформацію для подальшого аналізу продуктивності моделі.

```python
if span_id and span_id != "None" and span_id.strip():
    try:
        phoenix_host = os.getenv("PHOENIX_HOST", "http://localhost:6006")
        url = f"{phoenix_host}/v1/span_annotations?sync=false"

        annotation = {
            "span_id": span_id,
            "name": "user feedback",
            "annotator_kind": "HUMAN",
            "result": {
                "label": annotations.get("review"),
                "score": annotations.get("score"),
            }
        }

        response = requests.post(url, json={"data": [annotation]})
```

Рисунок 3.16 – Завантаження payload до системи Arize Phoenix



Рисунок 3.17 – Інтерфейс користувача для оцінки модуля

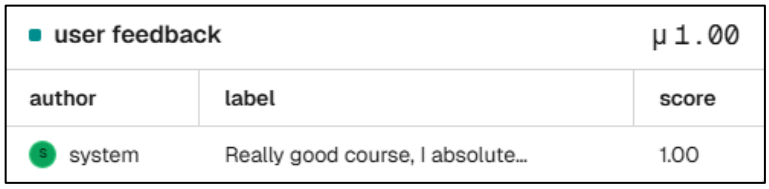


Рисунок 3.18 – Відображення коментаря в UI Arize Phoenix

3.5 Розробка сторінки технічної підтримки

Для визначення рівня володіння англійською мовою було запроваджено окремий модуль технічної підтримки. Технічна підтримка має можливість оцінювати завантажені сертифікати. Для цього була створена окрема функція, яка надає доступ до сторінки тоді і тільки тоді, коли користувач системи має роль «Technical Support» (див. рис. 3.19).

```
async def technical_support(self, request: Request):
    user_id = request.session.get("user_id")
    if not user_id:
        return RedirectResponse(url="/login", status_code=302)

    role = request.session.get("user_role")
    if not role:
        user = self.db.get_user_by_id(user_id)
        role = user.get("role") if user else None
        if role:
            request.session["user_role"] = role

    if role != "Technical Support":
        return RedirectResponse(url="/learn", status_code=302)

    return self.templates.TemplateResponse(request, "technical_support.html", {"request": request})
```

Рисунок 3.19 – Функція доступу до сторінки технічної підтримки

На рисунку 3.20 зображено основний функціонал сторінки. Співробітник технічної підтримки має можливість: переглянути всі сертифікати, які перебувають у статусі розгляду; перевірити наявність сертифікатів за певним email або user_id; за наявності сертифіката, після натискання кнопки «Access», у боковій панелі відображається інформація про користувача (поточний рівень володіння англійською мовою, ім’я та прізвище).

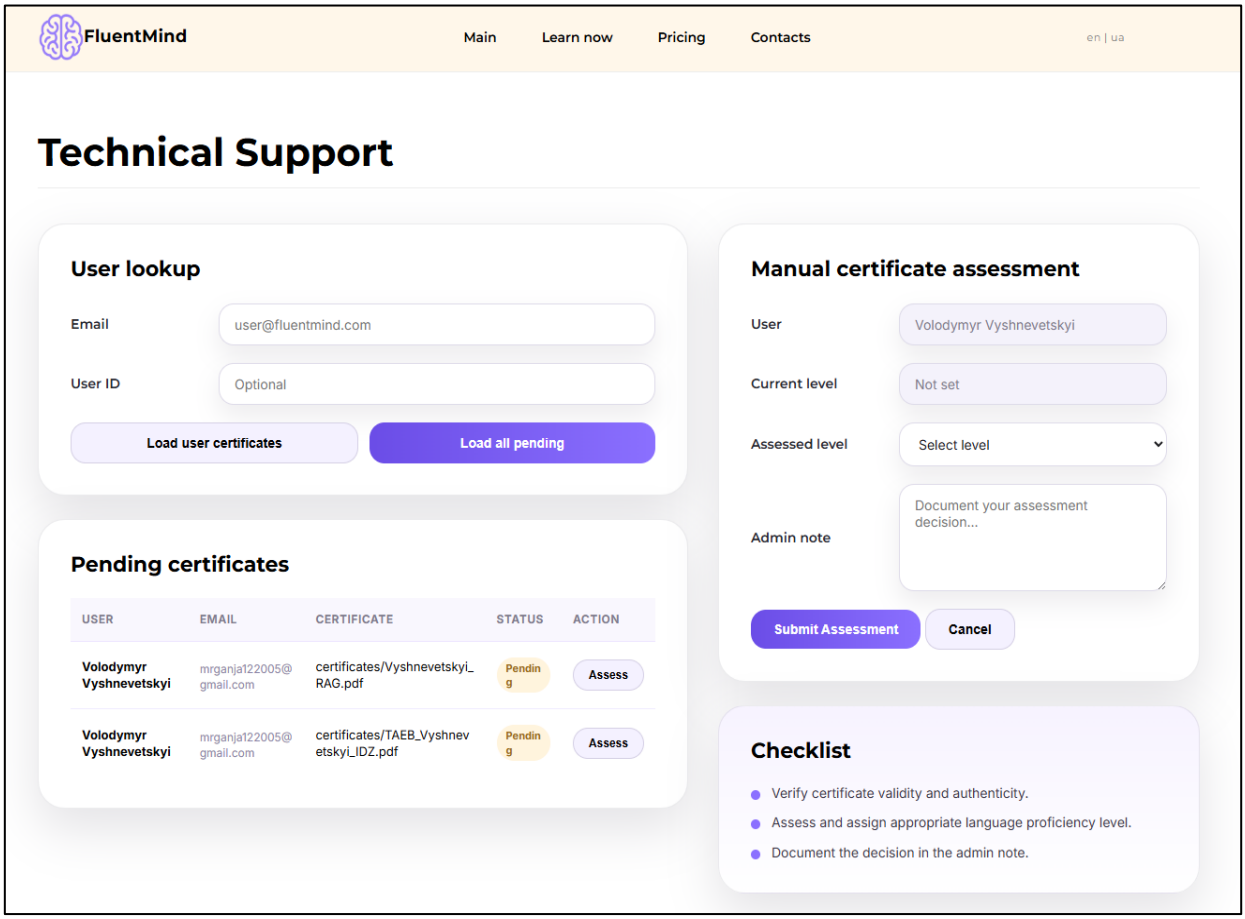


Рисунок 3.20 – Сторінка технічної підтримки

Співробітник технічної підтримки має встановити рівень володіння англійською мовою та надати коментар. Було реалізовано список основних правил, яких необхідно дотримуватися під час оцінювання сертифіката: перевірити автентичність сертифіката; встановити рівень англійської мови відповідно до сертифіката; додати відповідний коментар із прийнятим рішенням.

Для завантаження всіх сертифікатів, що очікують на перевірку, було створено функцію api_get_all_pending_certificates (див. рис. 3.21). Функція повертає JSON-об'єкт, що містить перелік шляхів до файлів. Використовується функція взаємодії з базою даних get_pending_certificates (див. рис. 3.22), яка знаходить та вибирає всі записи з таблиці сертифікатів, що ще не пройшли модерацію (мають статус 0, в базі даних).

```
async def api_get_all_pending_certificates(self, request: Request):

    admin_id = request.session.get("user_id")
    if not admin_id:
        return JSONResponse(status_code=401, content={"success": False, "message": "Not authenticated."})

    try:
        certificates = self.db.get_pending_certificates()
        return JSONResponse(status_code=200, content={
            "success": True,
            "certificates": certificates
        })
    except Exception as e:
        return JSONResponse(status_code=400, content={"success": False, "message": str(e)})
```

Рисунок 3.21 – Функція завантаження сертифікатів

```
def get_pending_certificates(self):
    connection = sqlite3.connect(self.db_name)
    cursor = connection.cursor()
    cursor.execute(
        '''
        SELECT c.certificate_id, c.user_id, c.certificate, c.status, u.name, u.surname, u.email
        FROM certificate c
        JOIN user u ON c.user_id = u.user_id
        WHERE c.status = 0
        '''
    )
    rows = cursor.fetchall()
    connection.close()
    certificates = []
    for row in rows:
        certificates.append({
            "certificate_id": row[0],
            "user_id": row[1],
            "certificate": row[2],
            "status": row[3],
            "user_name": row[4],
            "user_surname": row[5],
            "user_email": row[6]
        })
    return certificates
```

Рисунок 3.22 – Функція для відбору сертифікатів, що не пройшли модерацію

Таким чином, реалізований модуль технічної підтримки надає варіантивність та надійність системи оцінювання, автоматизує процес обробки сертифікатів та забезпечує коректне прийняття рішень співробітниками технічної підтримки.

ВИСНОВКИ

У кваліфікаційній роботі було розроблено додаток з вивчення іноземних мов із використанням таких засобів: Python, FastAPI, Arize Phoenix, LangChain, Weaviate, Docker, SQLite.

У першому розділі розглянуто предметну область та визначено вимоги до системи вивчення іноземної мови з інтеграцією генеративних моделей. Обґрунтовано актуальність теми в умовах широкого використання сервісів вивчення іноземних мов, таких як Duolingo, а також доцільність застосування машинного навчання для персоналізованого, адаптивного навчання. Сформовано коло потенційних користувачів системи, наведено загальні функціональні та нефункціональні вимоги, визначено основні задачі, які виконує система: оцінювання рівня CEFR, генерація курсів і модулів, надання зворотного зв’язку та підтримка навчального словника. Також було наведено огляд інструментів і бібліотек, що використовуються при розробці.

У другому розділі представлено етапи проєктування інформаційної системи, зокрема було розроблено концептуальну модель системи, що включає опис сценаріїв використання та акторів системи. Побудовано діаграму варіантів використання, у якій деталізовано функціональні потоки: від аутентифікації до генерації курсів. Описано логічну структуру системи, розроблено діаграму класів і фізичну модель: детально наведено компоненти системи, їх функціональні зв’язки та роль сторонніх бібліотек у побудові системи вивчення іноземних мов.

У третьому розділі наведено програмну реалізацію системи вивчення іноземної мови. Продемонстровано фрагменти коду, що відповідають за автентифікацію, роботу з профілем, генерацію тестів та курсів, перевірку відповідей, оцінку курсів, створення векторної бази даних та використання технік промпт-інженерії. Вебсервер реалізовано на FastAPI, інтерфейс користувача: HTML, CSS, JavaScript.

Подальший розвиток системи передбачає розширення механізмів персоналізації: запровадження спілкування між користувачем і LLM, додавання завдань з аудіювання, інтеграцію моделі з генерацією зображень для забезпечення глибшого занурення в освітній процес.

ПЕРЕЛІК ПОСИЛАНЬ

1. Duolingo Revenue and Usage Statistics. *Business of Apps*. URL: https://www.businessofapps.com/data/duolingo-statistics (дата звернення: 20.03.2026).

2. Team D. Introducing Duolingo Max, a learning experience powered by GPT-4. *Duolingo Blog*. URL: https://blog.duolingo.com/duolingo-max (дата звернення: 20.03.2026).

3. When Gamification Spoils Your Learning: A Qualitative Case Study of Gamification Misuse in a Language-Learning App / R. Hadi Mogavi and etc. *L@S '22* : Proceedings of the Ninth ACM Conference on Learning @ Scale. 2022. P. 175-188. URL: https://doi.org/10.1145/3491140.3528274 (дата звернення: 20.03.2026).

4. Live Roleplays powered by OpenAI Realtime API. *Speak*. URL: https://www.speak.com/blog/live-roleplays (дата звернення: 25.03.2026).

5. ChatGPT for good? On opportunities and challenges of large language models for education / E. Kasneci and etc. *Learning and Individual Differences*. 2023. Vol. 103. P. 102274. URL: https://doi.org/10.1016/j.lindif.2023.102274 (дата звернення: 27.03.2026).

6. Antal M., Buza K. Evaluating Open-Source LLMs in RAG Systems: A Benchmark on Diploma Theses Abstracts Using Ragas. *Acta Universitatis Sapientiae Inform*. 2025. Vol. 17, № 1. P. 1-15. URL: https://doi.org/10.1007/s44427-025-00006-3 (дата звернення: 27.03.2026).

7. General Python FAQ. *Python documentation*. URL: https://docs.python.org/3/faq/general.html#why-was-python-created-in-the-first-place (дата звернення: 27.03.2026).

8. Arena Leaderboard | Compare & Benchmark the Best Frontier AI Models. Arena | Benchmark & Compare the Best AI Models. URL: https://lmarena.ai/leaderboard/ (дата звернення: 27.03.2026).

9. Headway 5th edition Placement Tests. *Oxford University Press*. URL: https://elt.oup.com/teachers/headway/placement-tests/?cc=ua&selLanguage=uk&mode=hub (дата звернення: 10.04.2026).

10. Oxford University Press. *OUP*. URL: https://www.oup.es/oxfordtestofenglish/ (дата звернення: 10.04.2026).

11. Heymans M. Guided Learning in Gemini: From answers to understanding. *Google*. URL: https://blog.google/products-and-platforms/products/education/guided-learning/ (дата звернення: 10.04.2026).

12. Gemini Developer API pricing | Gemini API | Google AI for Developers. *Google AI for Developers*. URL: https://ai.google.dev/gemini-api/docs/pricing?hl=en (дата звернення: 12.04.2026).

13. Pricing | OpenAI API. *OpenAI*. URL: https://platform.openai.com/docs/pricing (дата звернення: 19.04.2026).

14. Maurya K. K., Srivatsa K. V. A., Petukhova K., Kochmar E. Unifying AI Tutor Evaluation: An Evaluation Taxonomy for Pedagogical Ability Assessment of LLM-Powered AI Tutors. *NAACL 2025*. 2025. URL: https://arxiv.org/abs/2412.09416 (дата звернення: 20.04.2026).

15. Evaluating Tutoring Feedback Quality. *Google Gemini*. URL: https://gemini.google.com/share/51f529482946 (дата звернення: 20.04.2026).

16. English platform. *Figma*. URL: https://www.figma.com/design/DhhDHhSSSu00KkEsjqkNrD/English-platform?node-id=1-2&t=zaJFIJYGOj9zQ6ac-1 (дата звернення: 25.04.2026).

17. GitHub – openlanguageprofiles/olp-en-cefrj: Open Language Profiles – English profile datasets from CEFR-J. *GitHub*. URL: https://github.com/openlanguageprofiles/olp-en-cefrj (дата звернення: 20.05.2026).

18. Вишневецький В. М., Кривохата А. Г., Гавриш І. В. Розробка системи вивчення англійської мови із застосуванням великих мовних

моделей. *Таврійський науковий вісник. Серія: Технічні науки*. 2026. № 3. С. 27–41. URL: https://doi.org/10.32782/tnv-tech.2026.3.3 (дата звернення: 28.05.2026).

ДОДАТОК А

Модуль промпт-інженерії

А.1 Промпт для оцінки якості пояснень LLM в якості тьютора

You are an English tutor helping a student improve their writing.

Student task:

"Write a short paragraph (3–4 sentences) about your favorite hobby in the past tense."

Student response:

"I like play basketball when I was young. I am practice every day with my friends.

We win many games and feel very happy."

Instructions:

- Read the student’s response carefully.
- Help the student improve their English.
- Do NOT rewrite the paragraph for the student.
- Point out any mistakes you notice.
- Explain the mistakes clearly and give helpful guidance.
- Encourage the student and keep a supportive tone.

А.2 Промпт для генерації екзамену

You are an expert English exam creator. Generate a 30-question multiple-choice English placement test.

IMPORTANT: Return ONLY a valid JSON object. Do NOT include any introductory text, markdown formatting (like ```json), or explanations. The output must be parseable by JSON.parse().

The JSON structure MUST be:

```
{
  "questions": [
    {
      "id": 1,
      "question": "Question text here",
      "options": {
        "A": "Option A text",
        "B": "Option B text",
        "C": "Option C text",
        "D": "Option D text"
      }
    }
  ]
}
```

Ensure the questions cover a range of difficulty levels (A1 to C2) to assess proficiency accurately.

JSON ONLY.

А.3 Промпт для генерації курсу

You are an expert English course creator. Create a detailed 8-week English course

for a student at {level} level. IMPORTANT: Return ONLY a valid

JSON object.

Do NOT include any introductory text, markdown formatting (like ```json),

or explanations. The output must be parseable by JSON.parse(). The JSON structure MUST be:

```
{{
  "title": "Course Title",
  "description": "Brief description of the course",
  "duration_weeks": 8,
  "course_plan": [
    {{
      "module": 1,
      "topics": ["Topic 1", "Topic 2"],
      "objectives": ["Objective 1", "Objective 2"],
      "activities": ["Activity 1", "Activity 2"]
    }}
  ]
}}
```

Ensure the course is engaging and covers all essential skills: reading, writing, speaking.

JSON ONLY.

A.4 Промпт для генерації зворотнього зв’язку

You are an expert English teacher grading a student's module submission.

Course ID: {course_id}

Module: {module_number}

Here are the exercises and the student's answers extracted from the page:

{exercises_text}

Please provide constructive feedback for the student.

1. Correct any mistakes politely.
2. Explain why an answer is wrong if it is.
3. Praise good answers.
4. Provide the correct answer if the student was wrong.

Format your response as HTML (just the inner body content, no <html> tags) suitable for displaying inside a div.

Use for each exercise feedback.

Add some styling classes.