НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих

комп'ютерних систем

До захисту допущено:

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою

«Системне програмування та спеціалізовані комп'ютерні системи»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Вебзастосунок для вивчення англійської мови із використанням штучного інтелекту»

Виконала:
студентка IV курсу, групи КВ-11
Петрук Ольга Сергіївна _____

Керівник:
доцент каф. СПіСКС, к. т. н.,
Петрашенко Андрій Васильович _____

Консультант з нормоконтролю:
доцент каф. СПіСКС, к. т. н.,
Клятченко Ярослав Михайлович _____

Рецензент:
доцент кафедри ПЗКС, к. т. н.,
Юрчишин Василь Якович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра системного програмування і
спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Системне програмування та спеціалізовані комп'ютерні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
«___»_____20__ р.

ЗАВДАННЯ
на дипломний проєкт студентки
Петрук Ольги Сергіївни

1. Тема проєкту «Вебзастосунок для вивчення англійської мови із використанням штучного інтелекту», керівник проєкту доцент каф. СПіСКС, к.т.н. Петрашенко А.В., затверджені наказом по університету від «29» травня 2025 р. №1808-С

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту: див. Технічне завдання

4. Зміст пояснювальної записки:

- Аналіз існуючих рішень та обґрунтування теми дипломного проекту.
- Вибір технологій і програмних засобів для реалізації вебдодатку.
- Тестування створеного вебдодатку

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

- Блок-схема логіки взаємодії Rasa з OpenAI
- Схема бази даних
- Архітектура взаємодії вебдодатку
- Блок-схема алгоритму роботи голосового AI-асистента
- Презентація дипломного проєкту

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., к. т. н., доцент каф. СПіСКС		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення літеоратури за тематикою проєкту	19.12.2024	
2	Розробка та узгодження технічного завдання	26.01.2025	
3	Аналіз існуючих рішень	01.03.2025	
4	Підготовка матеріалів першого розділу дипломного проєкту	12.04.2025	
5	Підготовка матеріалів другого розділу дипломного проєкту	07.05.2025	
6	Підготовка матеріалів третього розділу дипломного проєкту	17.05.2025	
7	Підготовка матеріалів четвертого розділу дипломного проєкту	22.05.2025	
8	Тестування розробленої системи	24.05.2025	
9	Оформлення документації дипломного проєкту	25.05.2025	
10	Попередній огляд матеріалів диплому на кафедрі	27.05.2025	
11	Захист	16.06.2025	

Студент Ольга ПЕТРУК

Керівник Андрій ПЕТРАШЕНКО

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (78 с., 74 рис., 1 табл., 4 додатки).

Об’єкт розробки – вебзастосунок для вивчення англійської мови, що використовує технології штучного інтелекту для індивідуального підбору навчальних вправ та інтерактивного спілкування.

Вебзастосунок дозволяє: сторювати навчальні модулі з різними іграми, генерувати різноманітні вправи з граматики, словникового запасу, аудіювання та читання відповідно до рівня користувача; здійснювати оцінювання відповідей із використанням семантичного аналізу; надавати інтерактивний мовний чат-бот для практики розмовної англійської з різними сценаріями; забезпечує розпізнавання мовлення за допомогою бібліотеки Vosk.

У ході розробки:

- проведено аналіз існуючих систем для вивчення англійської з ІІ, визначено їхні переваги та недоліки;
- сформульовані вимоги до вебзастосунку;
- розроблено вебзастосунок із використанням React, TypeScript, Node.js, OpenAI API, Rasa, Vosk та MongoDB;
- протестовано функціональність та якість навчання користувача.

У процесі розробки було використано: технології JavaScript (React, TypeScript), Node.js, API OpenAI, платформу Rasa для створення чат-бота, бібліотеку Vosk для розпізнавання мовлення, СУБД MongoDB для зберігання даних користувачів та навчальних матеріалів, а також REST API для інтеграції компонентів.

Ключові слова: ВЕБЗАСТОСУНОК, АНГЛІЙСЬКА МОВА, ШТУЧНИЙ ІНТЕЛЕКТ, НАВЧАЛЬНІ ВПРАВИ, СЕМАНТИЧНИЙ АНАЛІЗ, ЧАТ-БОТ, РОЗПІЗНАВАННЯ МОВЛЕННЯ, REACT, TYPESCRIPT, NODE.JS, OPENAI, RASA, VOSK, MONGODB.

ANNOTATION

The qualification work includes an explanatory note (78 pages, 74 figures, 1 table, 4 appendices).

The object of development is a web application for learning English that uses artificial intelligence technologies for personalized selection of learning exercises and interactive communication.

The web application enables the creation of learning modules with various games, generation of diverse exercises in grammar, vocabulary, listening, and reading according to the user's level; evaluation of answers using semantic analysis; provision of an interactive speech chatbot for practicing conversational English with different scenarios; and speech recognition using the Vosk library.

During development:

- an analysis of existing AI-based English learning systems was conducted, their advantages and disadvantages identified;
- requirements for the web application were formulated;
- the web application was developed using React, TypeScript, Node.js, OpenAI API, Rasa, Vosk, and MongoDB;
- functionality and quality of user learning were tested.

The development utilized: JavaScript technologies (React, TypeScript), Node.js, OpenAI API, Rasa platform for chatbot creation, Vosk library for speech recognition, MongoDB DBMS for storing user data and learning materials, as well as REST API for integrating components.

Keywords: WEB APPLICATION, ENGLISH LANGUAGE, ARTIFICIAL INTELLIGENCE, LEARNING EXERCISES, SEMANTIC ANALYSIS, CHATBOT, SPEECH RECOGNITION, REACT, TYPESCRIPT, NODE.JS, OPENAI, RASA, VOSK, MONGODB.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.002 ТЗ	Вебзастосунок для	4		
			вивчення англійської мови			
			із використанням			
			штучного інтелекту			
			Технічне завдання			
	А4	ІАЛЦ.045440.003 ТП	Вебзастосунок для	2		
			вивчення англійської мови			
			із використанням			
			штучного інтелекту			
			Відомість технічного			
			проєкту			
	А4	ІАЛЦ.045440.004 ПЗ	Вебзастосунок для	78		
			вивчення англійської мови			
			із використанням			
			штучного інтелекту			
			Пояснювальна записка			

Змін.	Арк.	№ докум.	Підпис	Дата
Розробила		Петрук О.С.		
Перевірив		Петрашенко А.В		
Консульт.				
Н. контроль		Клятченко Я.М.		
Зав. каф.		Романкевич В.О.		

ІАЛЦ.045440.001 ОА

Вебзастосунок для вивчення англійської мови із використанням штучного інтелекту
Опис альбому

Літ.	Аркуш	Аркушів
	1	3

КПІ
ім. Ігоря Сікорського,
ФПМ КВ-11

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.005 Д1	Вебзастосунок для	1		
			вивчення англійської мови			
			із використанням			
			штучного інтелекту			
			Блок-схема логіки			
			взаємодії Rasa з OpenAI			
	А4	ІАЛЦ.045440.006 Д2	Вебзастосунок для	1		
			вивчення англійської мови			
			із використанням			
			штучного інтелекту			
			Схема бази даних			
	А4	ІАЛЦ.045440.007 Д3	Вебзастосунок для	1		
			вивчення англійської мови			
			із використанням			
			штучного інтелекту			
			Архітектура взаємодії			
			вебдодатку			

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.008 Д4	Вебзастосунок для	1		
			вивчення англійської мови			
			із використанням			
			штучного інтелекту			
			Блок-схема алгоритму			
			роботи голосового			
			AI-асистента			
		Диск CD-ROM	Текст пояснювальної	1		
			записки. Графічний матеріал.			
			Фрагменти програмного			
			коду. Презентація			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.001 ОА	Арк. 3

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: Вебзастосунок для вивчення англійської мови із використанням штучного інтелекту.

Галузь застосування: Розробка призначена для користувачів, які хочуть ефективно покращити свої навички англійської мови шляхом індивідуального підбору навчальних вправ, інтерактивної практики мовлення та аудіювання з використанням сучасних технологій штучного інтелекту. Вебзастосунок може бути використаний у навчальних закладах та мовних школах, а також для самостійного навчання.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання для виконання роботи першого (бакалаврського) рівня вищої освіти, яке було затверджено кафедрою СПСКС Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою розробки є створення вебзастосунку для вивчення англійської мови з використанням штучного інтелекту, що забезпечує персоналізований підбір вправ, семантичну оцінку відповідей та інтерактивний голосовий чат-бот.

Призначення — надати зручний інструмент для користувачів різних рівнів, який сприятиме ефективному та самостійному навчанню.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки даного дипломного проекту є наукові публікації та статті, офіційні документації, науково-технічна література, електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

Вебзастосунок має виконувати основні функції:

- Реєстрація та автентифікація користувачів;
- Збереження та керування даними користувача у базі даних.
- Індивідуальний підбір навчальних вправ відповідно до рівня користувача;
- Генерація вправ з граматики, словникового запасу, аудіювання та читання;
- Оцінка відповідей з використанням семантичного аналізу;
- Інтерактивна мовна практика за допомогою чат-бота з різними сценаріями;
- Розпізнавання голосових команд користувача з використанням бібліотеки Vosk.

5.2. Вимоги до програмного забезпечення

- Операційна система Windows 10 або вище, Linux, MacOS;
- Наявність стабільного підключення до Інтернету;
- Ресурси для розробки: JavaScript (React, TypeScript), Node.js, Python.

5.3. Вимоги до апаратної частини

- Операційна система Windows 10 або вище, Linux, MacOS;
- Сучасний веб-браузер (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari);
- Доступ до Інтернету для взаємодії із сервером та API;
- Мікрофон для використання функцій розпізнавання мовлення.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів
1.	Вивчення літератури за тематикою проєкту	27.11.2024
2.	Розроблення та узгодження технічного завдання	20.01.2025
3.	Аналіз існуючих рішень	02.03.2025
4.	Підготовка матеріалів першого розділу дипломного проєкту	10.04.2025
5.	Підготовка матеріалів другого розділу дипломного проєкту	05.05.2025
6.	Підготовка матеріалів третього розділу дипломного проєкту	15.05.2025
7.	Підготовка матеріалів четвертого розділу дипломного проєкту	20.05.2025
8.	Тестування розробленої системи	25.05.2025
9.	Оформлення документації дипломного проєкту	26.05.2025
10.	Попередній огляд матеріалів диплому на кафедрі	27.05.2025

Пояснювальна записка
до дипломного проекту

на тему: Вебзастосунок для вивчення англійської мови із використанням штучного інтелекту

Київ – 2025

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

AI (Artificial Intelligence) – штучний інтелект

API (Application Programming Interface) – інтерфейс програмування додатків

DB (Database) – база даних

NLP (Natural Language Processing) – обробка природної мови

UI (User Interface) – користувацький інтерфейс

UX (User Experience) – користувацький досвід

Vosk – бібліотека для розпізнавання мовлення з відкритим кодом

Rasa – платформа для створення чат-ботів із використанням машинного навчання

React – JavaScript бібліотека для побудови користувацьких інтерфейсів

TypeScript – надмножина JavaScript із підтримкою статичної типізації

Node.js – серверна платформа для виконання JavaScript поза браузером

MongoDB – документно-орієнтована система керування базами даних

Семантичний аналіз – метод аналізу тексту для розуміння його змісту і значення

Чат-бот – програмний агент, що імітує спілкування з користувачем на природній мові

Аудіо транскрипція – процес перетворення голосового запису в текстовий формат

ВСТУП

У сучасному світі знання англійської мови є необхідністю для успішної комунікації, навчання та професійного розвитку тощо. Вивчення мови часто є складним та виснажливим процесом, особливо коли стандартні методи не враховують унікальність кожного учня. Кожна людина має власний темп навчання та інтереси, тому для досягнення справжнього успіху у вивченні потрібен персоналізований підхід.

Розвиток технологій штучного інтелекту відкриває нові можливості для створення систем, що можуть адаптуватися до індивідуальних потреб користувача, забезпечуючи інтерактивну та ефективну практику. Вебзастосунок, який поєднує інтелектуальні алгоритми та сучасні технології, може стати таким інструментом, що зробить навчання більш доступним, цікавим і результативним.

1 АНАЛІЗ ТЕМАТИКИ ТА ОБҐРУНТУВАННЯ ВИБОРУ ПРОЄКТУ

1.1 Актуальність теми

У теперішній час вивчення англійської мови залишається одним із найпопулярніших напрямів освіти у всьому світі. Зараз кожна людиа повинна знати англійську хоча б на базовому рівні для навчання, роботи, подорожей, спідкування та ще в багатьох сферах. Проте, рівень вивчення англійської мови у закладах освіти бажає бути кращим, а традиційні підходи часто не враховують індивідуальні особливості учнів, що знижує їх мотивацію, бажання та ефективність навчання. Але сучасні технології, серед яких використання штучного інтелекту, здатні покращити та змінити це. Інтерактивні чат-боти, адаптивні вправи, семантичний аналіз відповідей — це лише деякі з інструментів, які точно покращать якість та зручність навчання.

1.2 Цільова аудиторія

Цільовою аудиторією користувачів розробленого вебзастосунку є абсолютно всі люди, які зацікавлені у вивченні англійської мови. Користуватися цим сервісом можуть як початківці, так і просунуті та досвідчені учні, які прагнуть підвищити рівень володіння англійською мовою. Вебастосунок також створений для тих людей, які не мають можливості відвідувати офлайн курси чи мовні школи та практикувати розмовну англійську у зручному онлайн форматі.

1.3 Аналіз існуючих рішень

Наразі на ринку існує багато платформ для вивчення англійської мови, серед них Duolingo, Babbel, Rosetta Stone та інші. Всі вони пропонують структуровані курси, вправи та деякі інтерактивні елементи, але більшість із них не має гнучкої персоналізації на основі штучного інтелекту, а також обмежені в можливостях живого спілкування з ботом, що може розпізнавати та аналізувати усне мовлення.

Існуючі варіанти чат-ботів зачасту використовують або жорстко задані сценарії або мають обмежену здатність до розуміння природної мови.

Розглянемо деякі з них.

1.3.1 Duolingo

Першою і, напевно, найпопулярнішою платформою, яку слід розлянути, є Duolingo. Це мобільна платформа для вивчення мов, яка пропонує інтерактивні уроки у формі ігор із поступовим підвищенням складності.

Переваги:

- Великий вибір мов для вивчення;
- Безкоштовний базовий доступ із можливістю підписки для розширених функцій;
- Ігровий формат, який підвищує мотивацію користувачів;
- Регулярні оновлення.

Недоліки:

- Обмежена персоналізація навчання;
- Відсутність глибокого аналізу усних відповідей;
- Механічний підхід, що іноді не відповідає реальним мовним ситуаціям.

Отже, платформа Duolingo є хорошим вибором для початкового рівня, а особливо для дітей, завдяки своєму ігровому формату, що мотивує користувачів регулярно займатись. Але відсутність глибокої персоналізації та обмежена робота з усною мовою є проблемою для навчання для користувачів із середнім та вищим рівнем, які потребують більш складних завдань, індивідуального підходу та практики реальних мовних ситуацій.

1.3.2 Babbel

Другою платформою розглянемо Babbel. Вона орієнтована на розвиток практичних мовних навичок, зокрема розмовної англійської, через структуровані уроки й діалоги.

Переваги:

- Простий інтерфейс у використанні;
- Акцент на реальні ситуації та розмовні навички;
- Добре розроблені курси з поясненнями граматики.

Недоліки:

- Тільки платний доступ, без безкоштовного базового варіанту;
- Менше інтерактивності порівняно з ігровими платформами;
- Обмежені можливості для адаптації під індивідуальні потреби.

Отже, платформа Babbel пропонує хороше навчання мови з акцентом на її практичне використання, особливо у розмовних ситуаціях, що робить її корисним для тих користувачів, що прагнуть швидко та акцентовано розвивати саме розмовні навички. Однак відсутність безкоштовної версії та обмежена інтерактивність є очевидними мінусами для частини користувачів.

1.3.3 Rosetta Stone

Третьою платформою для розбору буде Rosetta Stone. Ця платформа на ринку є однією із найстаріших для вивчення мов. Вона використовує зображення, текст і звук для навчання слів та граматики шляхом повторення інтервалів без перекладу.

Переваги:

- Вбудоване розпізнавання мовлення для покращення вимови;
- Широкий вибір мов і тем.

Недоліки:

- Висока вартість підписки;
- Може бути складним для початківців через відсутність пояснень.

Отже, вивчення англійської мови, використовуючи платформу Rosetta Stone, дозволяє успішно формувати інтуїтиве розуміння мови та покращувати вимову завдяки використаним технологіям розпізнавання мовлення. Але висока

вартість і відсутність пояснень є вагомими недоліками, що можуть відвертати користувачів або просто ускладнити процес навчання, особливо початківцям. Також, через жорстку структуру курсів на цій платформі, немає можливості підлаштуватися під конкретні цілі або темп користувача.

ВИСНОВОК ДО РОЗДІЛУ 1

Отже, у результаті проведеного аналізу існуючих платформ для вивчення англійської мови, було констатовано наявність широкого спектру інструментів, що пропонують різні методи навчання, базуючись на інтерактивних вправах та розмовній практиці із чат-ботами. Проте більшість із таких сервісів мають обмежену адаптивність та не враховують індивідуальні особливості кожного користувача в достатньому обсязі, що знижує ефективність навчання. А розроблена інтеграція технологій штучного інтелекту, зокрема семантичного аналізу відповідей та голосового розпізнавання, взагалі є ледь не єдиним представником на ринку.

Отже, врахувавши вищесказані фактори, можна сказати, що розробка даного вебзастосунку є актуальною й перспективною. Очевидно, що ця розробка є кращою за конкурентів, бо поєднує такі інноваційні підходи. Вона точно сприятиме підвищенню якості вивчення англійської мови, забезпечуючи гнучкість та зручність для кожного користувача індивідуально.

2 ІНСТРУМЕНТИ ДЛЯ РОЗРОБКИ ВЕБЗАСТОСУНКУ

2.1 Вибір мови програмування

Вибір мови програмування є дуже важливим рішенням у процесі розробки будь-якого програмного забезпечення, оскільки від нього залежить ефективність реалізації, зручність та підтримка кінцевого продукту. А особливо важливим цей вибір стає коли створюються вебзастосунки, що використовують технології штучного інтелекту, де необхідно інтегрувати різноманітні компоненти, такі як клієнтська частина, серверна логіка та спеціалізовані сервіси обробки мови і тощо.

При виборі мови програмування для реалізації клієнтської частини важливо було врахувати потужну підтримку сучасних веб-технологій, швидкість розробки, гнучкість у створенні інтерфейсів користувача та легкість інтеграції із серверною частиною. А вже для серверної частини дуже важлива продуктивність, можливість роботи з різними API, підтримка асинхронної обробки запитів та наявність бібліотек для роботи зі штучним інтелектом і обробкою природної мови.

Для розробки даного вебзастосунку для клієнтської частини було обрано JavaScript із використанням бібліотеки React і мови TypeScript, що забезпечує типізацію та підвищує надійність написаного коду. Серверна частина реалізована на Node.js, що є достойним вибором для побудови масштабованих та швидкодіючих вебзастосунків завдяки асинхронній моделі обробки й великій кількості підтримуваних модулів.

Окремо слід сказати про використання мови Python для розробки спеціалізованих додаткових сервісів. Це Vosk та Rasa. Перший слугує для розпізнавання мовлення та транскрибування, а другий слугує для спілкування із за визначеними сценаріями. Вибір саме мови Python обгрунтований тим, що ці бібліотеки мають найкращу підтримку та реалізовані саме на цій мові, що робить її незамінною для компонентів штучного інтелекту в системі.

Отже, вибір мов програмування для розробки вебдодатку обгрунтований як з точки зору технічних можливостей, так і з огляду на специфіку завдань, що стоять на меті у продукту. Цей підхід дозволяє забезпечити зручність, високу якість та ефективність розробкленого продукту.

2.1.1 JavaScript та TypeScript

Для створення клієнтської частини вебзастосунку було обрано мову програмування JavaScript. Ця мова є дуже популярною та універсальною, що дозволяє реалізувати інтерактивність та динамічність на вебсторінках. Також вона підтримується всіма сучасними браузерами, що є дуже важливим. JavaScript — це кросплатформна об’єктно-орієнтована мова сценаріїв, яка використовується для надання інтерактивності вебсторінкам (наприклад, для створення складних анімацій, клікабельних кнопок, спливаючих меню тощо). Існують також більш розвинені серверні версії JavaScript, такі як Node.js, які дозволяють додавати більше функціональності до вебсайту, ніж просто завантаження файлів (наприклад, реальний час співпраці між кількома комп’ютерами). Усередині хост-середовища (наприклад, веббраузера) JavaScript може бути під’єднаний до об’єктів цього середовища для надання програмного контролю над ними.

JavaScript містить стандартну бібліотеку об'єктів, таких як Array, Map та Math, а також базовий набір елементів мови, як-от оператори, керуючі конструкції та інструкції. Базовий JavaScript можна розширити для різних цілей, доповнивши його додатковими об’єктами. Наприклад:

Клієнтський JavaScript розширює основну мову, надаючи об'єкти для керування браузером та його Об'єктною Моделлю Документа (DOM). Наприклад, клієнтські розширення дозволяють застосунку розміщувати елементи у HTML-формі та реагувати на події користувача, як-от натискання миші, введення у форму чи навігацію сторінками.

Серверний JavaScript розширює основну мову, надаючи об’єкти, що стосуються виконання JavaScript на сервері. Наприклад, серверні розширення дозволяють застосунку спілкуватися з базою даних, зберігати інформацію між сеансами виконання або працювати з файлами на сервері.

Це означає, що в браузері JavaScript може змінювати зовнішній вигляд вебсторінки (DOM), а JavaScript у Node.js на сервері може реагувати на спеціальні запити, які надсилає код, що виконується у браузері [1].

Проте, щоб підвищити якість та зрозумілість коду, а також для його кращої підтримки може застосовуватися TypeScript. Це така надмножина JavaScript, яка додає статичну типізацію. Використання цієї мови дає змогу виявляти помилки ще на етапі компіляції та уникати логічних помилок у процесі розробки.

Веббраузери — не єдині платформи, на яких використовується JavaScript. Деякі бази даних, такі як MongoDB та CouchDB, використовують JavaScript як мову сценаріїв та запитів. Кілька платформ для програмування на робочому столі та серверах, зокрема проект Node.js, надають середовище для програмування JavaScript поза браузером [2].

При розробці клієнтської частини вебдодатку було застосовано бібліотеку React, що дозволяє будувати інтерфейси із компонентів — ізольованих, повторно використовуваних блоків, які відповідають за певну частину інтерфейсу. Використання TypeScript у проектах на React забезпечує суворе визначення типів пропсів і станів компонентів, що значно підвищує стабільність застосунку.

Завдяки зв’язці React та TypeScript забезпечується плавна навігація, швидкий відгук інтерфейсу та адаптивність для різних пристроїв.

2.1.2 Node.js

Для написання серверної логіки вебзастосунку було використано Node.js. Це середовище виконання JavaScript поза браузером, яке ґрунтується на V8 рушії від Google Chrome. Це особливо підходить для побудови високопродуктивних,

масштабованих мережевих додатків завдяки своїй асинхронній, неблокуючій моделі вводу або виводу.

У створеному вебдодатку Node.js відповідає за обробку HTTP-запитів від клієнтської частини, логіку аутентифікації користувачів, роботу з базою даних MongoDB, управління сесіями користувачів, а також координацію взаємодії з сервісами ШІ. Через Node.js реалізовано API, яке приймає запити користувача, передає їх далі до ШІ-сервісів і повертає відповіді назад.

Отже, використання Node.js сприяє швидкій розробці завдяки великій кількості готових модулів (пакетів npm), активній спільноті та простоті масштабування.

2.1.3 Python

Python — це мова програмування загального призначення, яка використовується в широкому спектрі прикладних областей. У Python все є об'єктом і може оброблятися відповідно. Синтаксис Python розроблений для того, щоб бути читабельним і чистим, з акцентом на простоту і мінімалізм [9].

Для реалізації компонентів, пов’язаних із ШІ, було обрано мову Python, яка має велику кількість спеціалізованих бібліотек і фреймворків, що найкраще підходить для обробки природної мови (NLP), а також робить її ідеальним вибором для задач, що пов’язані загалом із ШІ.

У проекті Python використовується для двох основних сервісів Vosk та Rasa.

Vosk — бібліотека для розпізнавання мовлення, яка забезпечує перетворення аудіосигналу у текст у режимі реального часу. Vosk підтримує кілька мов, має високу точність і може працювати офлайн, що дозволяє зменшити затримки і підвищити конфіденційність користувачів. Vosk — це набір інструментів для розпізнавання мовлення. Найкращі особливості Vosk: підтримує понад 20 мов і діалектів: англійська, індійська англійська, німецька,

французька, іспанська, португальська, китайська, російська, турецька, в'єтнамська, італійська, нідерландська, каталанська, арабська, грецька, фарсі, філіппінська, українська, казахська, шведська, японська, есперанто, гінді, чеська, польська, узбецька, корейська, бретонська, гуджараті, таджицька, телугу. І список постійно поповнюється; працює офлайн, навіть на легких пристроях — Raspberry Pi, Android, iOS; встановлюється простою командою: pip3 install vosk; портативні мовні моделі для кожної мови важать лише близько 50 МБ, але для серверного використання доступні й більші моделі; має потоковий API для найкращого користувацького досвіду (на відміну від популярних Python-бібліотек для розпізнавання мовлення); має прив'язки до різних мов програмування — Java, C#, JavaScript тощо; дозволяє швидко змінювати словник для досягнення максимальної точності; підтримує ідентифікацію мовця поряд зі звичайним розпізнаванням мовлення [8].

Rasa — платформа для розробки інтелектуальних чат-ботів із функціями розуміння природної мови, управління діалогами і навчання на базі прикладів. Rasa дозволяє створювати кастомізовані сценарії спілкування, адаптовані до різних мовних рівнів і потреб користувачів. Rasa Open Source — це відкрита платформа для створення розмовного штучного інтелекту, яка дозволяє розуміти та вести діалоги, а також підключатися до месенджерів і сторонніх систем через набір API. Вона надає будівельні блоки для створення віртуальних (цифрових) помічників або чат-ботів [6].

Ці сервіси працюють як незалежні окремі мікросервіси, що запускаються на Python і взаємодіють з основним Node.js сервером через API.

2.1.4 Переваги використання комбінованого технологічного стеку

Обраний підхід із розподілом функцій між різними мовами програмування базується на максимальному використанні переваг кожної технології:

- JavaScript/TypeScript (React) забезпечує зручний, швидкий та адаптивний інтерфейс користувача;

- Node.js дає можливість швидко й ефективно обробляти мережеві запити, керувати логікою застосунку і з’єднувати різні сервіси;
- Python із Vosk і Rasa відповідає за розпізнавання мовлення та інтелектуальну обробку природної мови.

Таке поєднання гарантує високу продуктивність системи, гнучкість у розвитку функціоналу та простоту інтеграції нових технологій у майбутньому.

2.2 Вибір фреймворку

Для розробки клієнтської частини вебдодатку було обрано бібліотеку React від компанії Facebook, яка на сьогодні є однією з найпопулярніших технологій для створення складних інтерфейсів користувача. React.js — це популярна JavaScript-бібліотека з відкритим кодом, розроблена та підтримувана компаніями Facebook і Instagram. Вона призначена для створення швидких, інтерактивних і масштабованих користувацьких інтерфейсів (UI). React пропонує розробникам компонентний підхід, при якому інтерфейс розбивається на незалежні, багаторазово використовувані блоки — компоненти. Це дозволяє значно спростити організацію коду та підвищити продуктивність розробки. Справжня сила React полягає у використанні власних компонентів для побудови (та оновлення!) інтерфейсу додатка. Компоненти можуть приймати властивості та відображати або поводитися по-різному залежно від значеньвластивостей. React заохочує створення маленьких, повторно використовуваних компонентів, які можна компонувати для створення складних інтерфейсів користувача. Використовуючи JSX, можна писати HTML-подібний синтаксис безпосередньо у JavaScript-коді, що полегшує візуалізацію структури ваших UI-компонентів [4].

Основна функція React полягає у ефективному керуванні відображенням (рендерингом) вебзастосунку за допомогою віртуального DOM (внутрішньої копії структури сторінки, що мінімізує роботу з реальним DOM і підвищує швидкість оновлення інтерфейсу).

React-додатки складаються з компонентів. Компонент — це частина інтерфейсу користувача (UI), яка має власну логіку та зовнішній вигляд. Компонент може бути розміром з кнопку або розміром з цілу сторінку [3].

React підтримує два основних типи компонентів. Перший тип це функціональні компоненти. Вони приймають вхідні параметри, тобто props, та відповідають за вивід простого елементу інтерфейсу без управління станом.

Другий тип це класові компоненти, що можуть містити внутрішній стан, тобто state, та обробляти складнішу логіку, зокрема життєвий цикл компонента.

Також серед ключових переваг React можна виділити:

- Висока продуктивність завдяки віртуальному DOM;
- Зручний та модульний підхід з компонентами, що спрощує масштабування проєктів;
- Широка підтримка TypeScript для більшої безпеки коду;
- Величезна екосистема з додатковими бібліотеками для маршрутизації, управління станом і тестування;
- Активна спільнота розробників, яка постійно розвиває і підтримує технологію;
- Можливість створення не лише вебдодатків, але й мобільних (React Native) та настільних (Electron) застосунків.

До недоліків можна віднести:

- Відсутність готових рішень, які присутні у повноцінних фреймворках — React орієнтований виключно на UI;
- Потреба у виборі додаткових бібліотек для управління станом і маршрутизації.

React.js широко застосовується у відомих світових проєктах, таких як Facebook, Instagram, Netflix, Uber, Airbnb, WhatsApp та інших. Це свідчить про його надійність і гнучкість для розробки як невеликих вебсайтів, так і великих масштабних систем.

Отже, вибір React.js як основна бібліотека для фронтенд-розробки вебзастосунку обумовлений його високою продуктивністю, гнучкістю, модульністю та широкою підтримкою в спільноті.

2.3 Вибір бази даних

Критично важливим етапом розробки вебзастосунку є вибір бази даних, оскільки це забезпечує надійне зберігання, доступність та гнучку обробку даних. Для створеного продукту важливо було обрати СУБД, що поєднує масштабованість, продуктивність та зручність розробки, тому чудово підійшла MongoDB — документоорієнтована NoSQL СУБД.

MongoDB Atlas — це повністю керована хмарна база даних, побудована на документній моделі, яка є найшвидшим способом для інновацій, оскільки документи набагато простіші та природніші у використанні. Ви можете зберігати дані будь-якої структури та змінювати схему в будь-який момент, додаючи нові функції до своїх застосунків [10].

Серед переваг MongoDB можна виділити наступні основні. Першою є гнучка модель даних, це значить, що MongoDB зберігає інформацію у вигляді документів JSON-подібної структури BSON, що дозволяє зберігати складні дані без необхідності точного та строгого визначення схеми. Також перевагою є висока продуктивність MongoDB, оскільки вона оптимізована для швидкого запису і читання даних, що є особливо важливим для реального часу, наприклад, під час роботи чат-бота або під час запису результатів тестів користувачів. Також суттєвою перевагою є масштабованість, завдяки якій MongoDB може розподіляти навантаження між кількома серверами, що дозволяє забезпечити стабільну роботу навіть при збільшенні кількості користувачів і обсягів даних. Для сворепного вебдодатку дуже важлива простота інтеграції з JavaScript та Node.js. Оскільки серверна частина проєкту написана на Node.js, MongoDB ідеально підходить завдяки природній підтримці JSON та зручним драйверам,

таким як Mongoose, які полегшують роботу з базою даних та дозволяють швидко реалізовувати бізнес-логіку. А також MongoDB має активну спільноту та широку екосистему, багато документації та готових рішень.

Необходиться і без недоліків, ось наступні можливі. Першим недоліком може бути високий обсяг споживання пам’яті, оскільки документоорієнтована модель вимагає більше ресурсів для зберігання даних порівняно з оптимізованими реляційними таблицями. Другим недоліком є менша підтримка складних запитів, тобто для деяких складних аналітичних завдань MongoDB може бути менш ефективною, ніж реляційні СУБД. І також у MongoDB існує потреба у правильному плануванні схеми даних, адже незважаючи на гнучкість, неправильне проектування схеми даних може призвести до проблем із продуктивністю.

Отже, загалом MongoDB є достойним вибором для розробленого вебзастосунку завдяки своїй гнучкості, продуктивності та зручності інтеграції з JavaScript-стеком.

2.4 Вибір середовища для розробки

Не менш важливим аспектом у розробці класного вебзастосунку є вибір середовища розробки для ефективного та зручного процесу. Таке середовище має забезпечувати комфортну роботу розробнику, підтримувати необхідні інструменти для написання, тестування та налагодження коду, а також інтегруватися з обраними технологіями.

Основними критеріями вибору середовища розробки стали:

Підтримка обраних мов програмування та фреймворків, тобто середовище має бути сумісним з JavaScript, TypeScript, Node.js, а також з Python, оскільки бекенд у проєкті реалізований на Node.js, а сервіси розпізнавання мовлення (Vosk) та чат-бота (Rasa) на Python.

Інструменти для ефективного написання коду, оскільки важливо, щоб

середовище пропонувало інтелектуальне підсвічування синтаксису, автодоповнення, перевірку помилок у режимі реального часу та швидку навігацію по коду.

Отже, для розробки вебдодатку використовується середовище розробки Visual Studio Code. Він є популярним редактор коду з відкритим вихідним кодом, який має багатий набір розширень для підтримки JavaScript, TypeScript, React, Node.js та Python. VS Code відзначається швидкою роботою, гнучкістю налаштувань та активною спільнотою, що забезпечує постійне оновлення та актуальність.

Плюси використання VS Code:

- безкоштовний та з відкритим вихідним кодом;
- легкий та швидкий;
- легко налаштовується за допомогою розширень;
- інтегрований Git та інструменти для налагодження;
- крос-платформна підтримка (Windows, macOS, Linux).

Мінуси використання VS Code:

- може бути затратним по ресурсам при встановленні багатьох розширень;
- бракує деяких розширених функцій повноцінних IDE, таких як Visual Studio або JetBrains IntelliJ;
- вимагає ручного налаштування для деяких мов програмування.

Отже, вибір Visual Studio Code, як середовища для розробки вебдодаток, повністю відповідає вимогам проєкту, оскільки забезпечує зручний та ефективний функціонал для роботи з різними технологіями, що використовуються у розробці продукту.

ВИСНОВОК ДО РОЗДІЛУ 2

Під час роботи було проведено комплексний аналіз основних технологій та інструментів для розробки вебзастосунку для вивчення англійської мови із використанням штучного інтелекту. Було чітко та повністю пояснено вибір мов програмування, де для основної серверної частини було використано Node.js, завдяки його високій продуктивності, асинхронній архітектурі та широкій екосистемі пакетів, що значно спрощує створення масштабованих вебсервісів. Для додаткових сервісів, пов'язаних з обробкою природної мови і розпізнаванням мовлення, було вибрано мову програмування Python, оскільки бібліотеки Vosk та Rasa найкраще підтримуються саме цією мовою, що гарантує їх стабільну, чисту та ефективну роботу.

Для клієнтської частини вебдодатку, щоб створити гарний, зрозумілий та адаптивний користувацький інтерфейс, було вибрано React.js. Це бібліотека, що дозволяє створювати компонентну архітектуру, яка сприяє повторному використанню коду, а також забезпечує високий рівень продуктивності за рахунок віртуального DOM та ефективного оновлення інтерфейсу. Також у цій бібліотеці є широка підтримка спільноти розробників та постійне оновлення, що підвищують її стабільність.

Для зберігання даних було обрано MongoDB. Ця система дуже добре підходить для зберігання гнучких, напівструктурованих даних, характерних для навчальних матеріалів та користувацьких налаштувань. Перевагами цієї системи є масштабованість, швидкість роботи з великими обсягами даних і простота інтеграції з Node.js.

Як середовище розробки було обрано Visual Studio Code. Це середовище ідеальне для комфортної та ефективної розробки вебдодатку. VS Code підтримує широкий спектр мов та технологій, має багатий набір розширень, та інструменти для контролю версій, що в купі сприяє прискоренню процесу розробки та зменшенню помилок.

Отже, загалом, вибір стеку технологій для розробки вебдодатку був здійснений на основі аналізу їх функціональних можливостей, сумісності між собою, а також відповідності потребам проєкту.

3 РОЗРОБКА ВЕБДОДАТКУ

3.1 Розробка структури додатка

Для створення вебзастосунку для вивчення англійської мови із використанням штучного інтелекту було обрано сучасний та потужний стек технологій. Для клієнтської частини (інтерфейс, з яким взаємодіє користувач) було використано React.js і TypeScript, що дозволяють створювати динамічні, швидкі та адаптивні інтерфейси. Ця частина забезпечує відображення навчальних модулів, інтерактивних вправ, чат-бота та інших компонентів.

Для серверної частини (логіка роботи застосунку, що обробляє запити від клієнта, взаємодіє з базою даних, керує генерацією вправ та інтегрується з сервісами штучного інтелекту) було використано Node.js, завдяки його асинхронності, широкій екосистемі пакетів та зручності масштабування. Окремі сервіси для розпізнавання мовлення (Vosk) та обробки природної мови (Rasa) реалізовані на Python, оскільки ці бібліотеки мають найкращу підтримку саме у цій мові.

Як базу даних (основна система зберігання даних користувачів, навчальних матеріалів, результатів виконання вправ та налаштувань) було обрано MongoDB, що є гнучкою NoSQL базою даних, здатною ефективно обробляти великі обсяги даних і легко інтегрується з Node.js.

Розробка вебдодатка починається зі створення основних компонентів інтерфейсу, які мають бути зрозумілими для користувача.

Архітектурно додаток складається з таких основних модулів:

Навігаційні / Вступні сторінки:

- Home (Головна сторінка) – це перша стоірнка після аутентифікації, де користувач бачить доступні навчальні ігри та вправи.
- Hello (Авторизаційна сторінка) – це перша і єдина сторінка, яка доступна для неаутентифікованих користувачів, яка дозволяє користувачу обрати між реєстрацією та входом до системи.

Основні функціональні модулі:

- Cards – це модуль створення та редагування навчальних карток з підтримкою drag-and-drop для зручного управління картками.
- Chat – це інтерактивний чат із підтримкою голосових повідомлень та текстового вводу, який поєднує роботу Rasa та OpenAI.
- Dictionary – це особистий словник користувача із можливістю додавання нових слів, перегляду та перевірки перекладу, а також ігровим режимом для практики.
- Exercises – це модуль для проходження різних типів вправ (граматика, словниковий запас, слухання, читання), з оцінкою та зворотним зв’язком.
- Library – це модуль для зберігання навчальних модулів із можливістю їх перегляду.
- ListeningPractice – це модуль для практики сприйняття англійської мови на слух із регулюванням складності, акценту та швидкості відтворення.
- ModulePage – це більш детальна сторінка конкретного навчального модуля з різними режимами навчання: картки, запам’ятовування, тестування та вимова.
- TongueTwisterChallenge – це модуль, що у розважальному форматі, служить для тренування вимови через складні скоромовки з оцінкою якості вимови на основі аудіозапису.
- VocabularyBuilder – це генератор вправ на заповнення пропусків у реченнях для покращення словникового запасу із перевіркою відповідей.
- AISpeakingBuddy – це модуль для практики мовлення з використанням аудіозаписів та ШІ, що розпізнає аудіо, транскрибує та відповідає користувачу.

- WordExplorer – це модуль для вивчення слів за категоріями, із генерацією нових слів ШІ, їх озвученням, прикладами та можливістю додавання у словник користувача.

Взаємодія між цими модулями реалізована через REST API, які забезпечують обмін даними між клієнтською та серверною частинами і додатковими сервісами штучного інтелекту.

Початковою сторінкою є Hello, на якій користувач має можливість вибору логіну або реєстрації.

Після успішної реєстрації або автентифікації користувач потрапляє на сторінку Home. Ця сторінка є головною та містить доступ до різноманітних навчальних модулів і вправ. Вона включає в себе панель з навігацією, яка розташована з лівого боку екрана і дає доступ до таких сторінок: Home, Library, Cards, Exercises, Chat, Dictionary. Також з цієї сторінки можна перейти на такі ігри як AISpeakingBuddy, WordExplorer, ListeningPractice, TongueTwisterChallenge та VocabularyBuilder, про які було написано вище.

Загалом, обрана структура вебдодатка забезпечує зрозумілий та зручний інтерфейс для користувача.

3.2 Блок-схема логіки взаємодії Rasa з OpenAI

На рис. 3.1 зображено блок-схему логіки обробки текстового запиту за допомогою системи Rasa і за потреби із можливістю виклику OpenAI.

Згідно схеми алгоритму, починається все з текстового запиту користувача. Цей запит надходить до модуля Rasa, який виконує аналіз повідомлення та визначає намір користувача.

Далі система переходить до умовного блоку, де перевіряється, чи належить намір до категорії FAQ (визначних запитань). Якщо так, то формується локальна відповідь із заздалегідь визначеного набору відповідей у системі, а якщо ні, то запит передається до OpenAI, який генерує динамічну відповідь за допомогою

великої мовної моделі GPT.

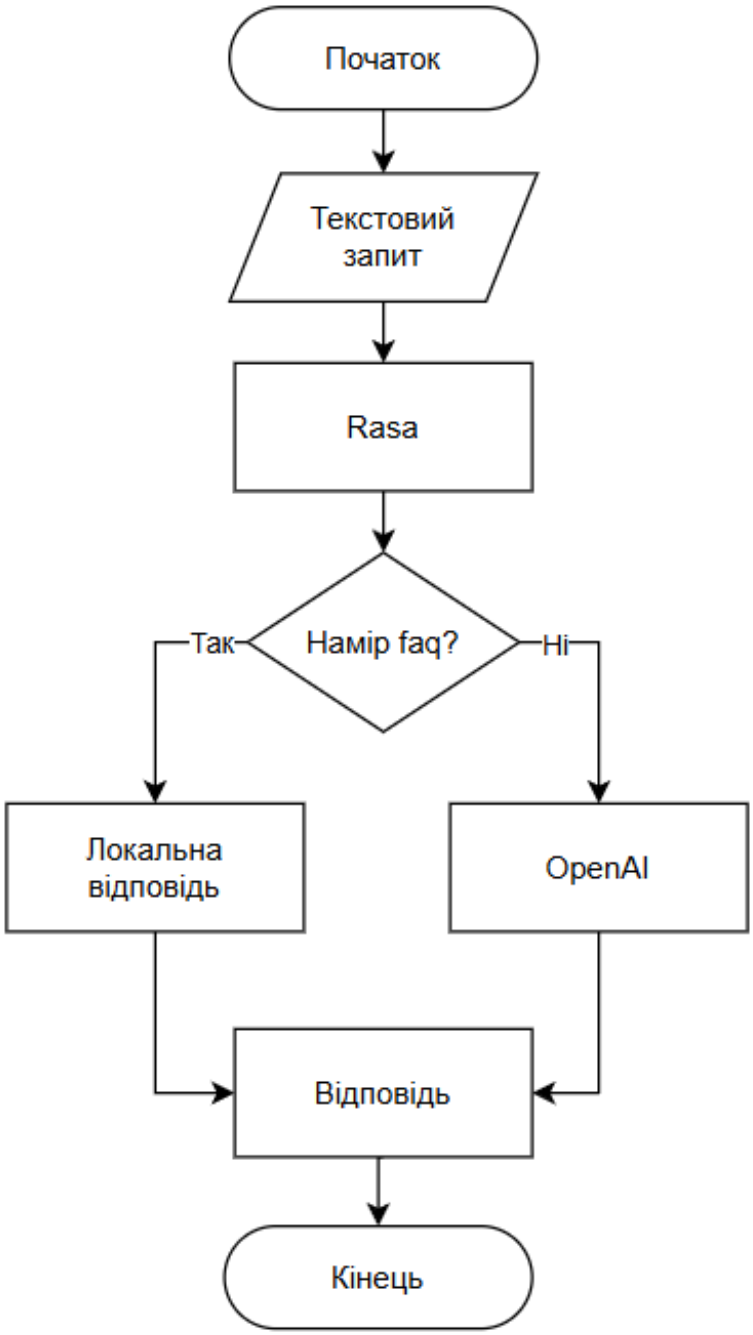


Рисунок 3.1 – Блок-схема логіки взаємодії Rasa з OpenAI

У підсумку обидва шляхи (локальна відповідь або OpenAI) об'єднуються у спільний блок, який означає видачу відповіді.

Ця розроблена схема ілюструє гібридну модель обробки запитів, де визначені запити обробляються локально сервером Rasa, а невизначені обробляються за допомогою ChatGPT.

3.3 Розробка схем бази даних

Дуже важливим етапом розробки додатку є створення схем бази даних, оскільки вони визначають структуру збереження та взаємодії з даними. Кожна схема описує набір полів та типів даних, які зберігаються в колекціях MongoDB.

Розглянемо основні створені колекції, їхні поля та взаємодію між собою.

Колекція Users зберігає інформацію про користувачів системи. Кожен користувач може мати багато модулів та слів у своєму словнику. Ось основні поля цієї колекції:

- _id – унікальний ідентифікатор користувача, тип: string;
- username – ім’я користувача, тип: string;
- email – електронна пошта користувача, тип: string;
- avatarPath – URL до аватара користувача, тип: string | null;
- googleId – унікальний ідентифікатор користувача в Google, тип: string | null;
- accessToken – токен доступу для аутентифікації, тип: string | null;
- refreshToken – токен оновлення Google, тип: string | null;
- subscriptionType – тип підписки користувача (наприклад, "free" або "premium"), тип: string;
- password – хешований пароль користувача (для локальної аутентифікації), тип: string | null;
- subscriptionExpiry – дата закінчення підписки, тип: Date | null;
- modules – масив ідентифікаторів модулів, що належать користувачу, тип: [ObjectId] (посилання на модулі);
- dictionary – масив ідентифікаторів слів, які зберігаються у словнику користувача, тип: [ObjectId] (посилання на слова).

Рисунок 3.2 – Приклад схеми User

Колекція Modules зберігає навчальні модулі, що містять групи навчальних карток. Модулі належать певним користувачам. Ось основні поля цієї колекції:

- _id – унікальний ідентифікатор модуля, тип: string;
- name – назва навчального модуля, тип: string;
- description – опис модуля, тип: string | null;
- cards – масив навчальних карток, тип: Array<object> (кожен елемент містить термін і визначення);
- createdAt – дата створення модуля, тип: Date;
- updatedAt – дата останнього оновлення модуля, тип: Date;
- user – ідентифікатор користувача, до якого належить цей модуль, тип: ObjectId (посилання на User).

Рисунок 3.3 – Приклад схеми Module

Колекція Words зберігає слова для словника користувача з перекладами, транскрипцією та прикладами. Кожен користувач може додавати слова в свій словник. Ось основні поля цієї колекції:

- _id – унікальний ідентифікатор слова, тип: string;
- word – назва слова на англійській мові, тип: string;
- translation – переклад слова на українську мову, тип: string;
- transcription – фонетична транскрипція, тип: string;
- synonyms – список синонімів для слова, тип: Array<string>;
- example – приклад речення з використанням слова, тип: string | null;
- user – ідентифікатор користувача, до якого належить це слово, тип: ObjectId (посилання на User).

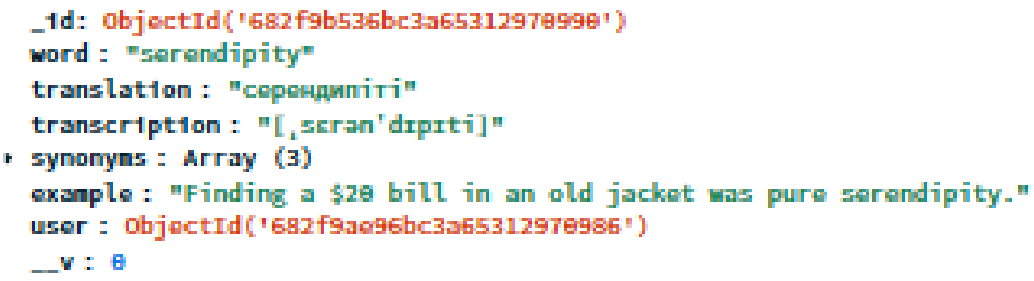


Рисунок 3.4 – Приклад схеми Word

На рис. 3.5 можна побачити повну схему бази даних.

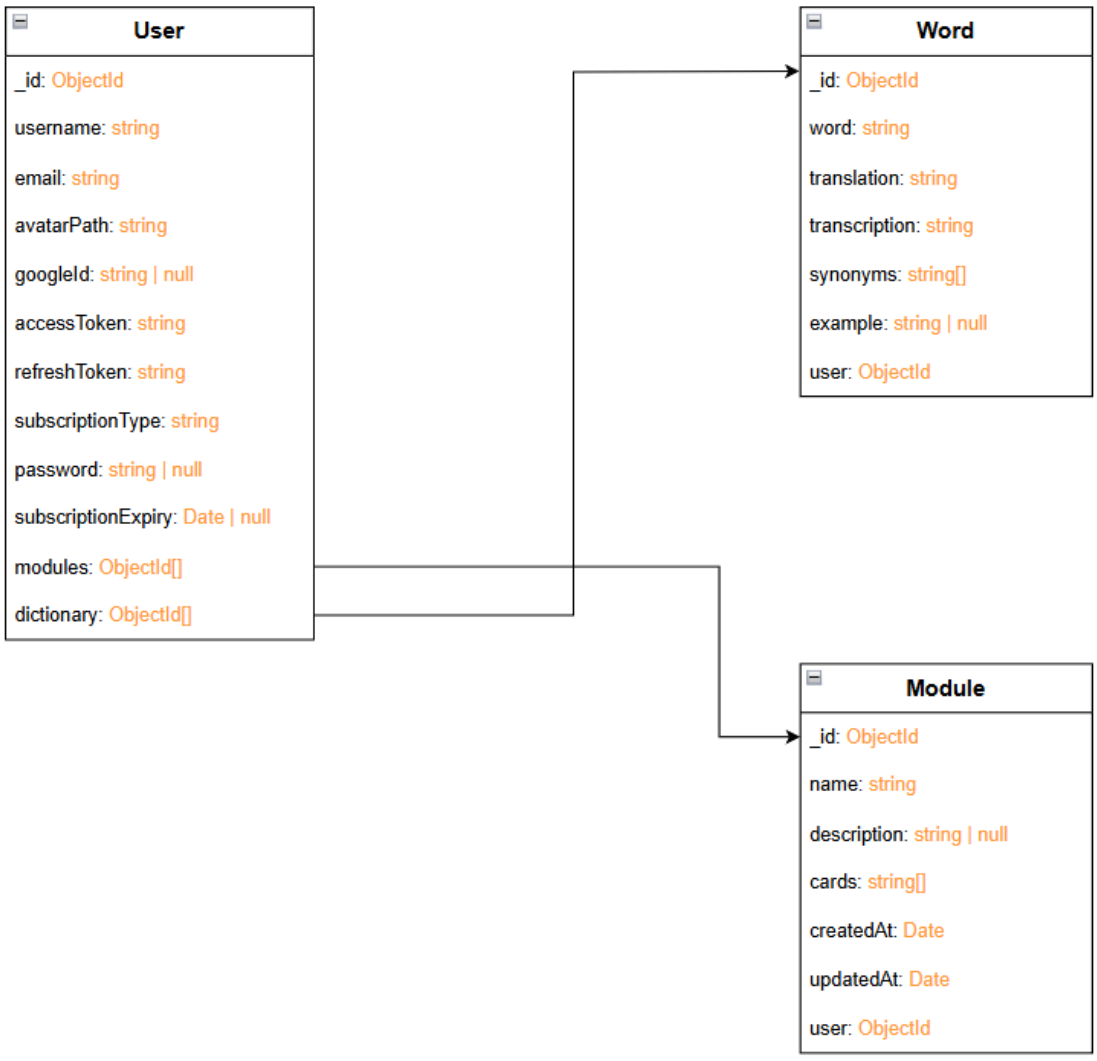


Рисунок 3.5 – Схема бази даних

3.4 Архітектура взаємодії вебдодатку

На рис. 3.6 показано схему архітектури взаємодії вебдодатку із зовнішніми сервісами та клієнтами.

У центрі схеми розташовано Центральний сервер обробки запитів, який виконує роль основного вузла координації між усіма модулями системи. Він забезпечує маршрутизацію даних, обробку запитів користувачів та взаємодію із зовнішніми та внутрішніми сервісами.

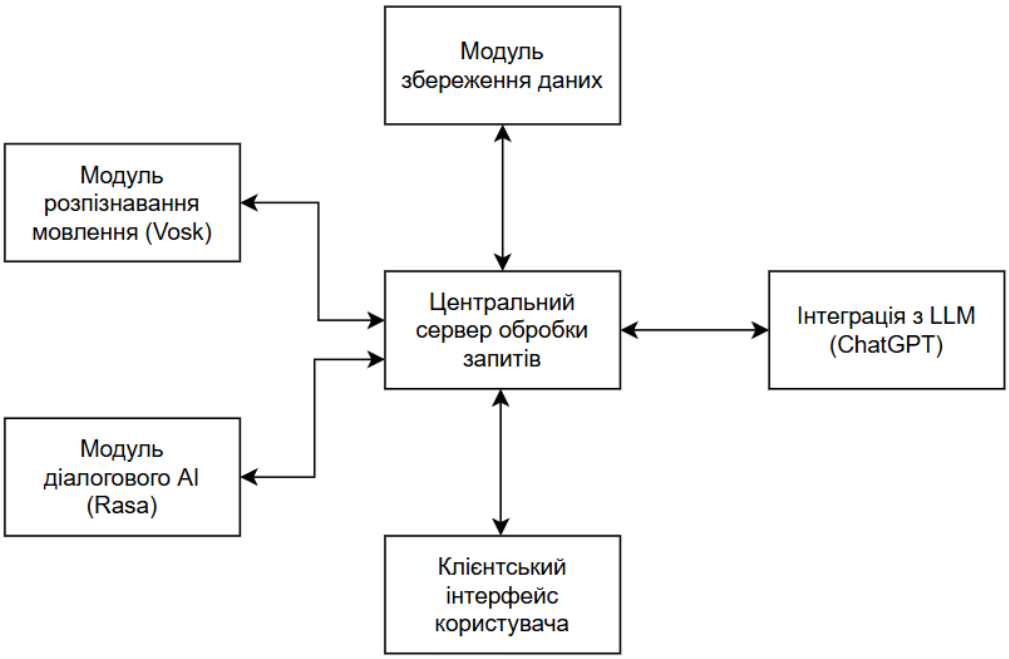


Рисунок 3.6 – Схема архітектури взаємодії вебдодатку

Модуль збереження даних відповідає за зчитування та запис інформації про користувачів, їх модулі та слова. Сервер надсилає запити на читання та зберігання до цього модуля при кожній сесії користувача.

Інтеграція з LLM (ChatGPT) реалізує генерацію відповідей у випадках, коли інші компоненти не можуть самостійно згенерувати відповідь, або коли потрібно скласти завдання до вправ, переклад тощо. Сервер формує запит до API LLM і обробляє отриману відповідь.

Модуль розпізнавання мовлення (Vosk) обробляє вхідне голосове повідомлення користувача та перетворює його в текстову транскрипцію. Сервер передає аудіо в цей модуль, а далі використовує отриманий текст у діалоговій логіці.

Модуль діалогового AI (Rasa) виконує обробку природної мови, інтерпретує наміри користувача, генерує відповіді. Якщо Rasa не зможе дати корекну відповідь, сервер перенаправляє запит до LLM-модуля (ChatGPT) як

fallback-механізм.

Клієнтський інтерфейс користувача це точкою взаємодії користувача із системою. Надсилає запити різного формату до центрального сервера та отримує відповідь.

Отже, схема демонструє модульну побудову системи з чітким розмежуванням функціональності, де кожен компонент відповідає за свою частину обробки. Центральний сервер виступає комунікаційним хабом.

3.5 Розробка блок-схеми алгоритму роботи голосового AI-асистента

На рис. 3.7 показана блок-схема алгоритму обробки голосових даних, яка описує логіку роботи голосового інтерфейсу вебдодатку.

Спершу все починається із передачі голосового повідомлення користувачем, яке записується та надходить до модуля розпізнавання мови (Vosk). Цей модуль у свою чергу виконує транскрипцію, тобто перетворює аудіо в текстовий формат. Отриманий текст передається до модуля обробки природної мови (Rasa), де відбувається аналіз змісту та спроба сформувати відповідь.

У випадку, якщо Rasa не може згенерувати відповідь, запит автоматично надсилається до модуля генерації відповіді (OpenAI), де за допомогою великої мовної моделі генерується відповідна текстова відповідь. Після цього незалежно від джерела формування відповіді, вона надходить до модуля синтезу мовлення (TTS), де перетворюється у голосовий сигнал. У підсумку, користувач отримує аудіо-відповідь, що завершує один повний цикл взаємодії.

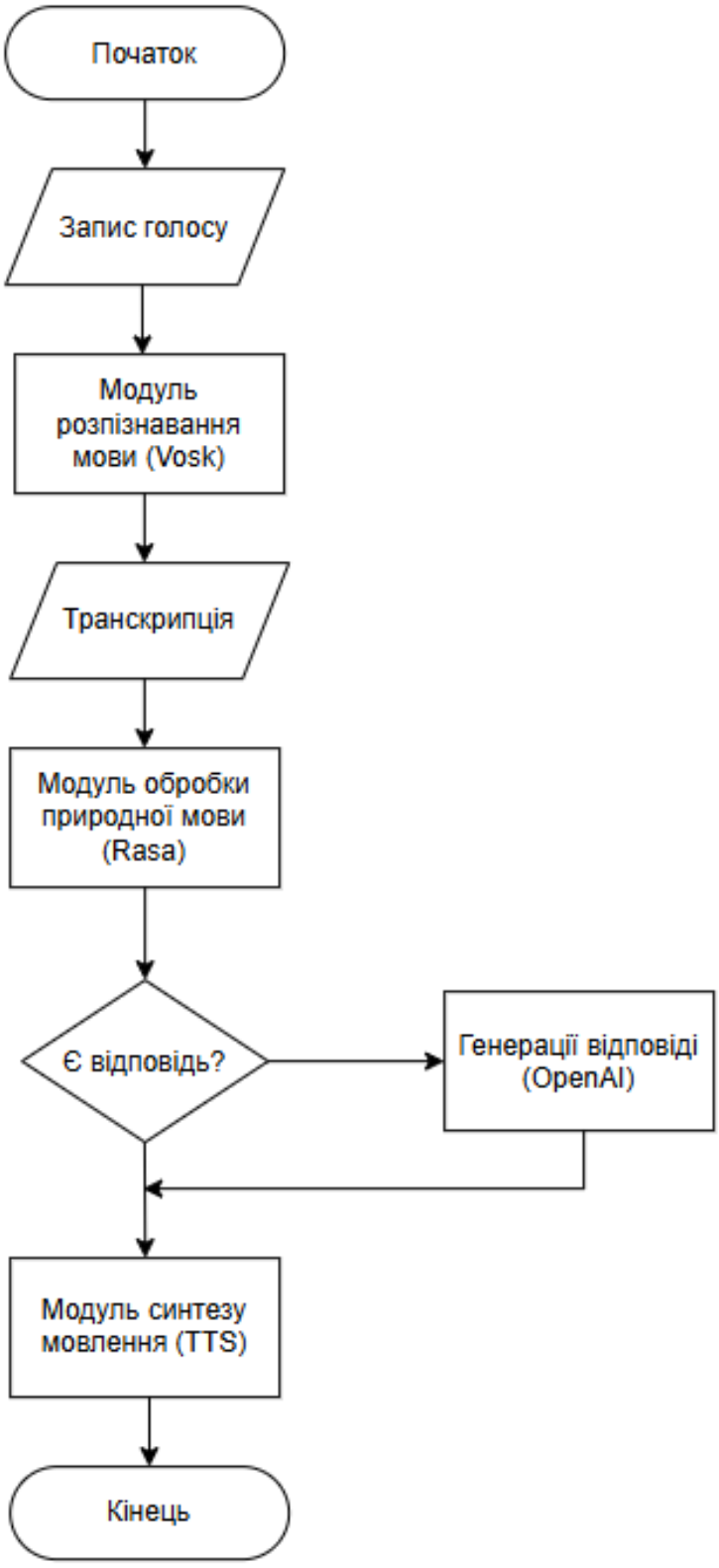


Рисунок 3.7 – Блок-схема алгоритму роботи голосового AI-асистента

ВИСНОВОК ДО РОЗДІЛУ 3

У результаті виконання було повністю реалізовано функціональну структуру створеного вебдодатку. Було добре спроєктовано і клієнтську і серверну частини системи, враховуючи зручнійсть, продуктивність та масштабованість використання. Реалізація була проведена із наступнийм стеком технологій: React, TypeScript, Node.js, MongoDB, використання яких дозволило створити гнучку архітектуру із можливістю подальшого розширення вебдодатку.

Користувачам надається великий вибір навчальних модулів, наприклад, інтерактивні ігри, картки, вправи, чат з AI-асистентом та словник. Особливістю системи є модулі з голосовою взаємодією, які базуються на інтеграції з сервісами Vosk (розпізнавання мовлення), Rasa (обробка природної мови) та OpenAI (генерація контенту та відповідей), що дозволяє забезпечити персоналізований та ефективний підхід до вивчення мови.

Було розроблено усі ключові сценарії взаємодії користувача із системою, включаючи авторизацію, використання навчальних модулів та отримання зворотного зв’язку було розроблено добре. Також додатково було розроблено структуру бази даних з основними колекціями, а саме колекції користувачі, словникові записи та модулі, що дозволяє надійно зберігати дані користувачів та забезпечувати їхню цілісність та безпеку.

Схеми архітектури взаємодії продемонструвала високий рівень модульності та взаємозв’язку між компонентами системи, що забезпечує ефективну роботу всіх сервісів. Розроблена система легко адаптується до нових функцій.

Отже, створена структура вебдодатку повністю відповідає таким поставленим завданням як створення гнучкого, зручного та функціонального інструменту для вивчення англійської мови з підтримкою голосових технологій та ШІ.

4 ТЕСТУВАННЯ ВЕБДОДАТКУ

4.1 Тестування інтерфейсу

Тестування інтерфейсу вебдодатку є важливим етапом, який забезпечує зручність та правильність взаємодії користувачів із системою у користуванні. Метою тестування інтерфейсу є виявлення помилок у відображенні елементів на сторінках, функціональності всіх видимих елементів, форм вводу та їх валідацію, навігації та адаптивності інтерфейсу на різних розмірах екрану.

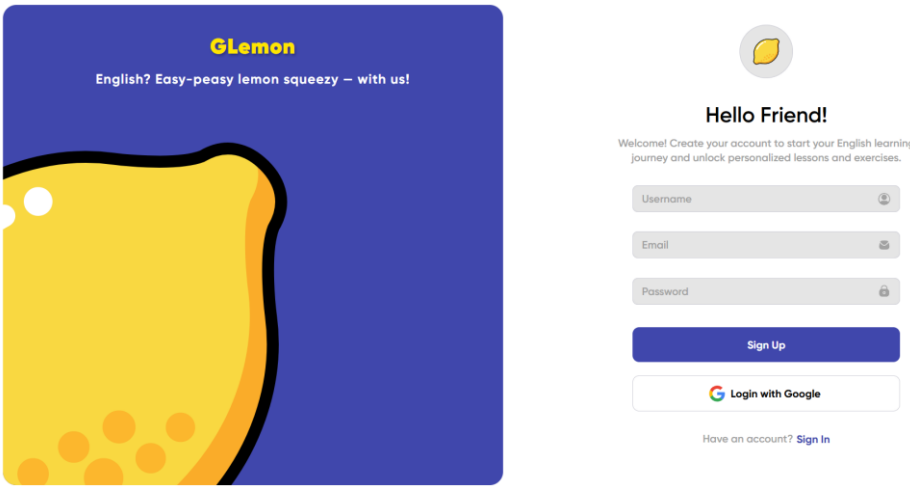


Рисунок 4.1 – Сторінка реєстрації

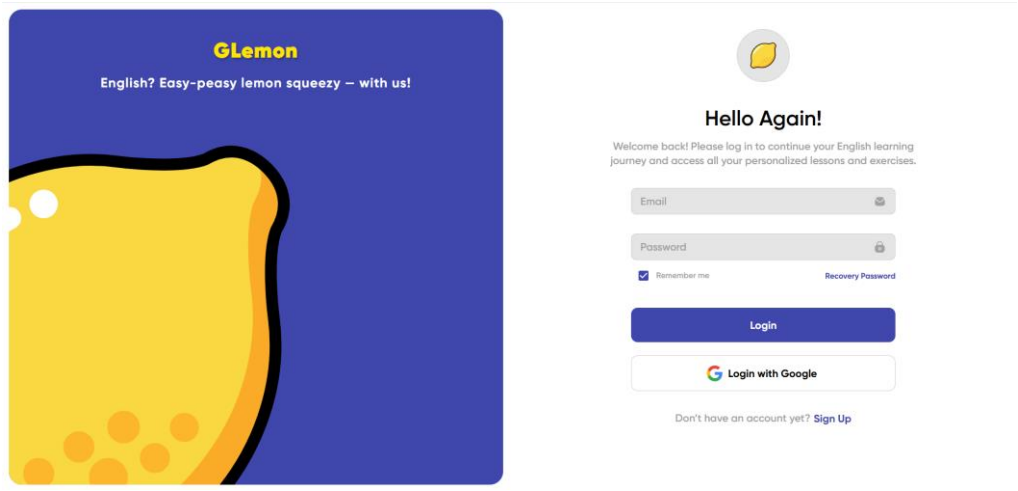


Рисунок 4.2 – Сторінка автентифікації

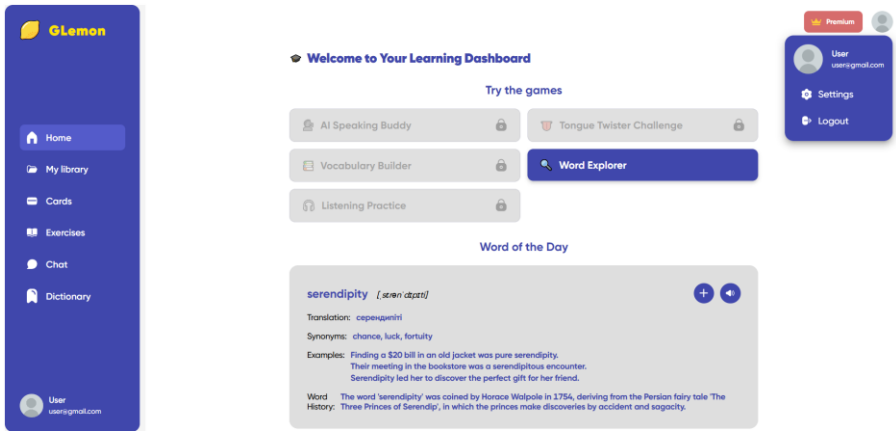


Рисунок 4.3 – Сторінка всередині вебдодатка, вказує на успішний вхід

4.2 Функціональне тестування

Функціональне тестування вебдодатку є дуже важливим етапом, оскільки саме тут перевіряються робота всіх функцій вебдодатку на відповідність вимогам і очікуванням користувача. Метою тестування є перевірка коректної обробки дій користувача, правильної взаємодії із основним сервером, базою даних та додатковими серверами для унікальних функцій.

Під час тестування вебдодатку має проводитися перевірка основних функціональних можливостей, а саме реєстрація та авторизація користувачів, управління навчальними модулями, створення та редагування навчальних карток, генерація та проходження різноманітних вправ, робота зі словником, а також інтеграція із сервісами штучного інтелекту для аналізу мовлення та перевірки відповідей.

Тестування має висвітлити як позитивні варіанти подій, коли користувач виконує дії у відповідності до інструкцій, так і негативні випадки, які перевіряють, як система реагує на некоректні дані. Також у додатку є платна можливість розширення підписки, яка знімає обмеження у функціоналі та яку теж треба перевірити.

Отже, у підсумку, функціональне тестування забезпечує впевненість у стабільності роботи вебдодатку.

4.2.1 Реєстрація користувача

Необхідно перевірити, чи коректно працює функціональність реєстрації користувачів у вебдодатку.

Очікується, що користувач може зареєструватися, використовуючи особисті дані (нікнейм, електронну пошту та пароль), і буде успішно збережений у базу даних та перенаправлений на головну сторінку.

Також очікується, що при помилкових даних, тобто при введенні невалідних даних (невалідна пошта або пароль) має спрацювати віладація на формі реєстрації.

Перевіримо правильність реєстрації користувача:

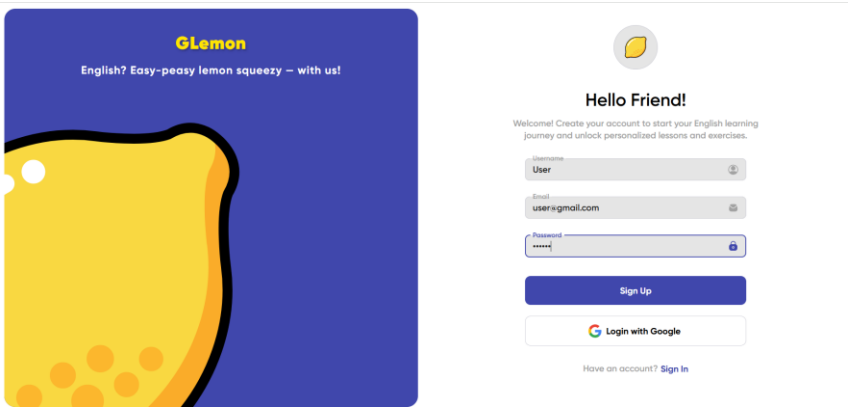


Рисунок 4.4 – Процес реєстрації нового користувача

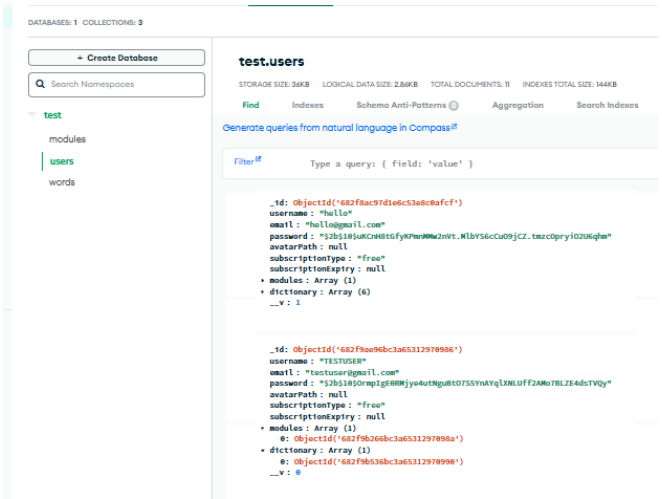


Рисунок 4.5 – Колекція Users до реєстрації нового користувача

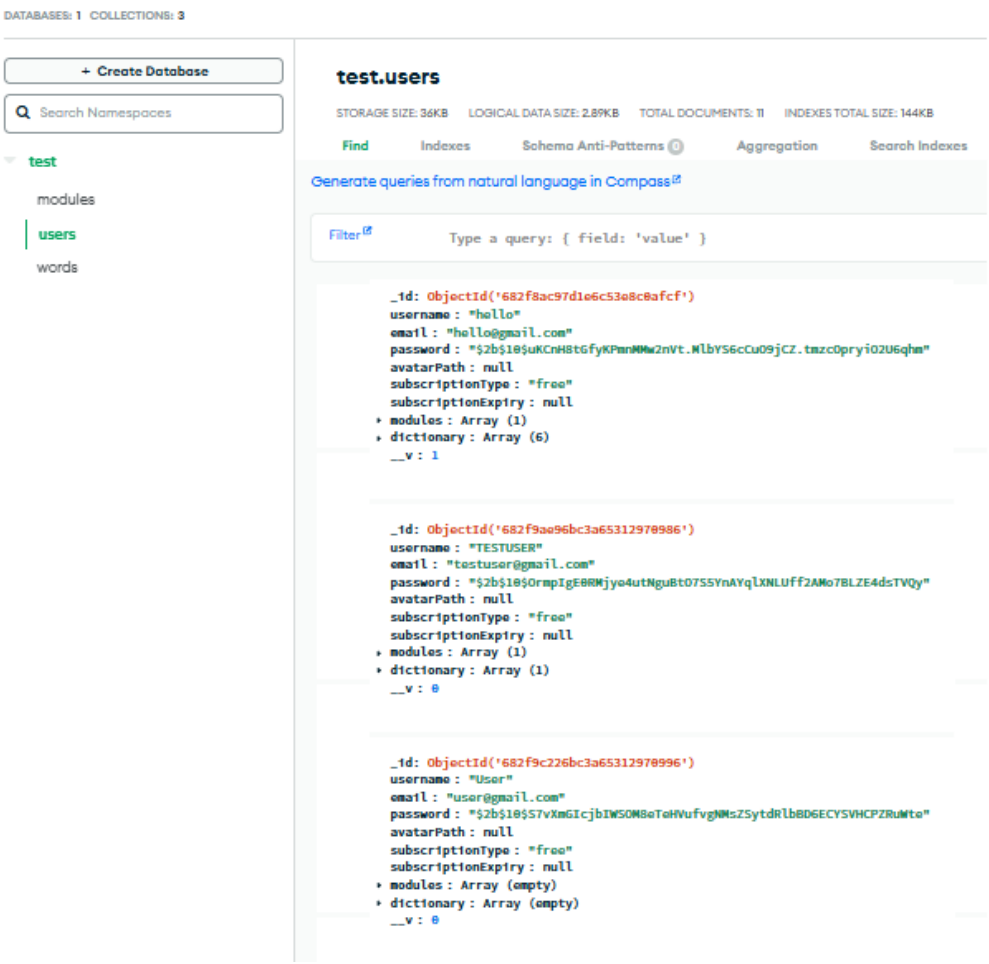


Рисунок 4.6 – Колекція Users після реєстрації нового користувача

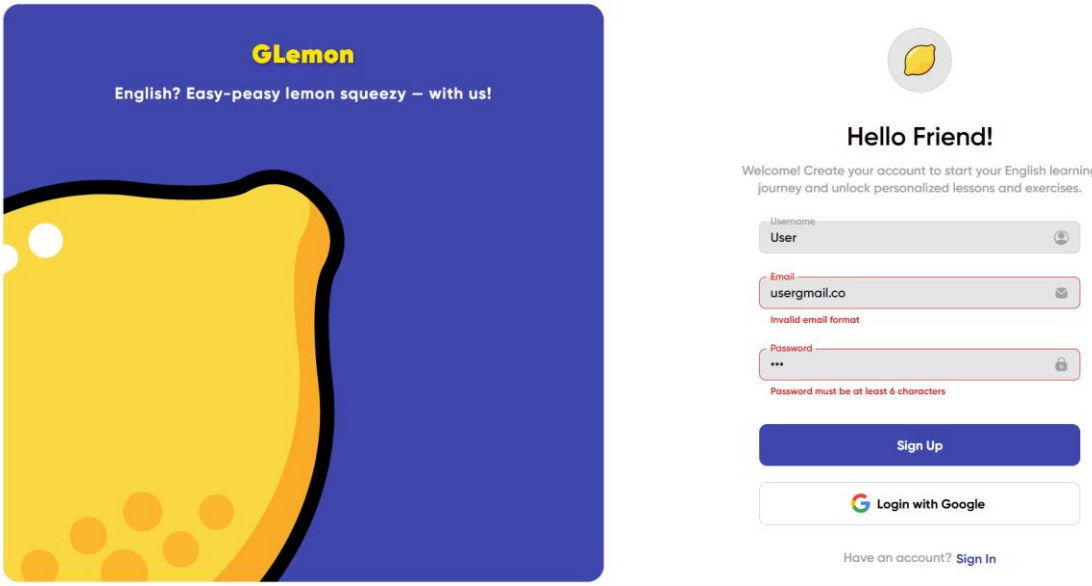


Рисунок 4.7 – Валідація форми реєстрації при помилкових форматах даних

Таким чином, на вищепоказаних скриншотах можна побачити, що при реєстрації все виконується правильно і дані внесені користувачем зберігаються в базу даних у колекцію Users. Перевірка була проведена за допомогою MongoDB Atlas. Це хмарна послуга, яка дозволяє використовувати базу даних MongoDB у хмарному середовищі.

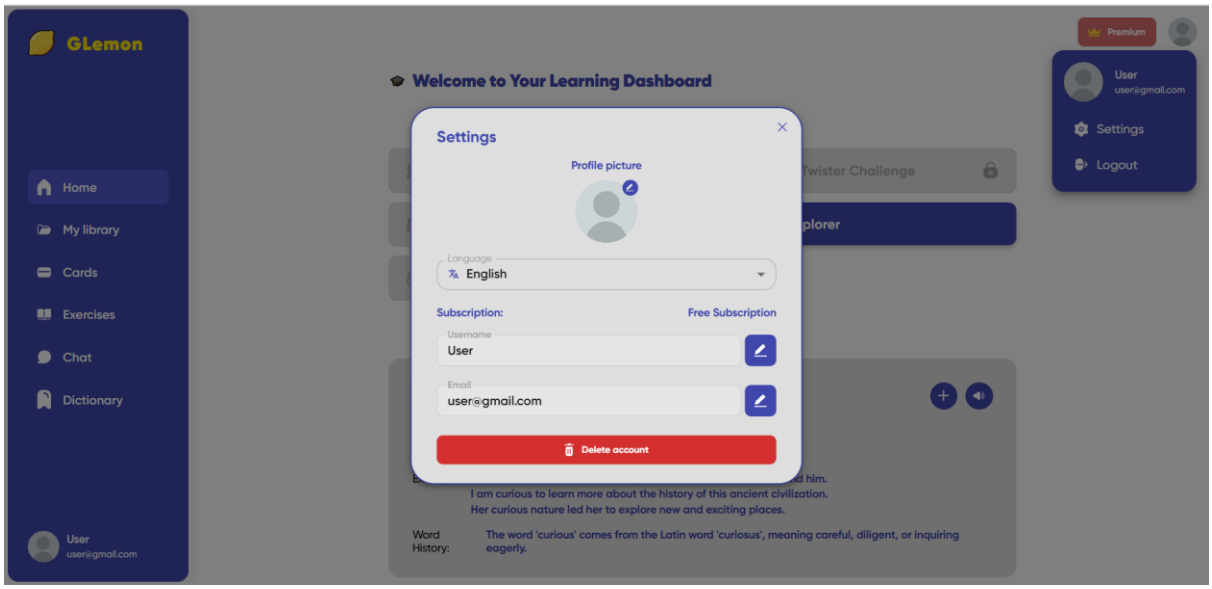


Рисунок 4.8 – Дані користувача на сторінці після реєстрації

4.2.2 Аутентифікація користувача

Необхідно перевірити, чи коректно працює функціональність входу для зареєстрованих користувачів.

Очікується, що користувач може увійти, використовуючи коректні дані (електронну пошту та пароль), і буде перенаправлений на головну сторінку.

Також очікується, що при помилкових даних, тобто при введенні неправильних даних (неіснуюча пошта або невірний пароль) система повинна вивести відповідне повідомлення про помилку.

Перевіримо правильність аутентифікації користувача:

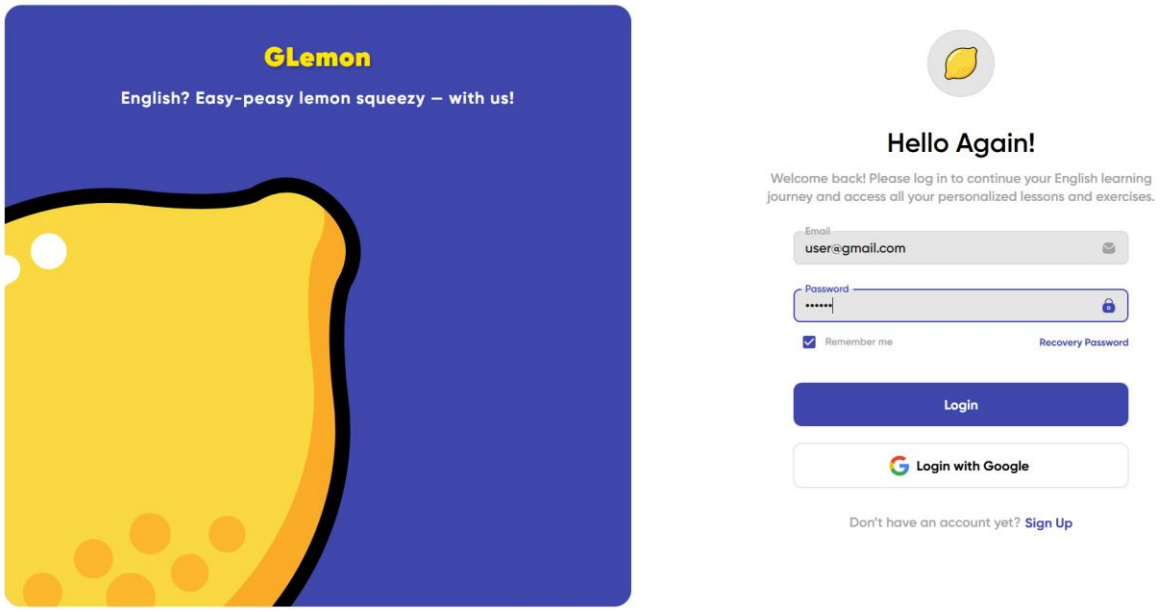


Рисунок 4.9 – Процес аутентифікації користувача

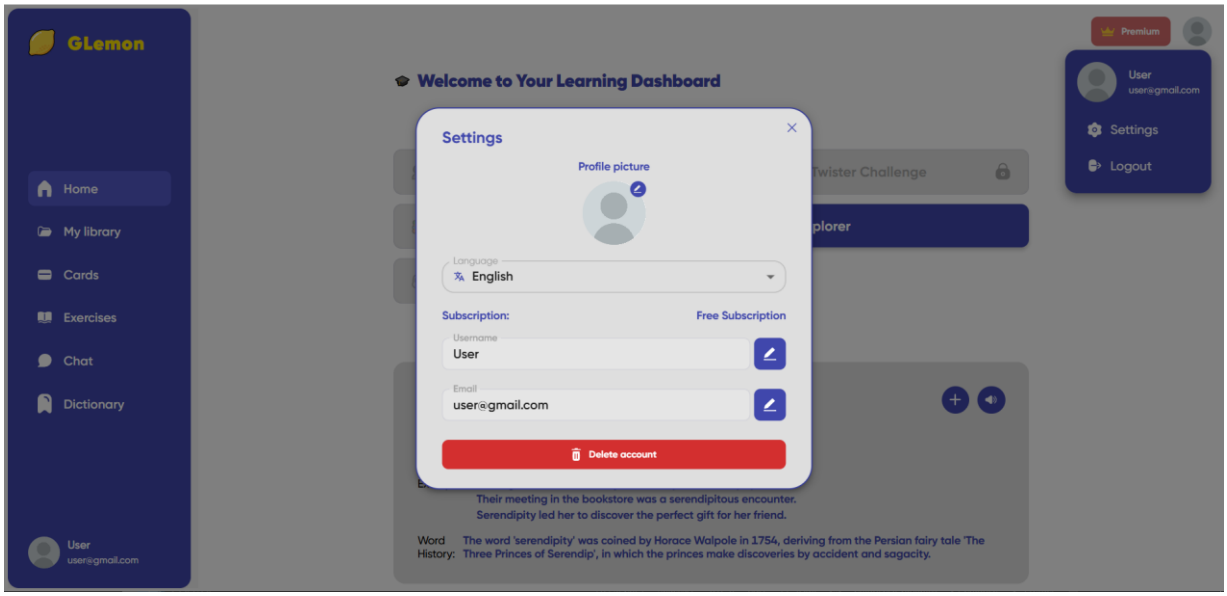


Рисунок 4.10 – Дані користувача на сторінці після успішної аутентифікації

У разі правильно введених даних ми перейдемо на головну сторінку сайту та побачимо основні дані користувача (рис. 4.10).

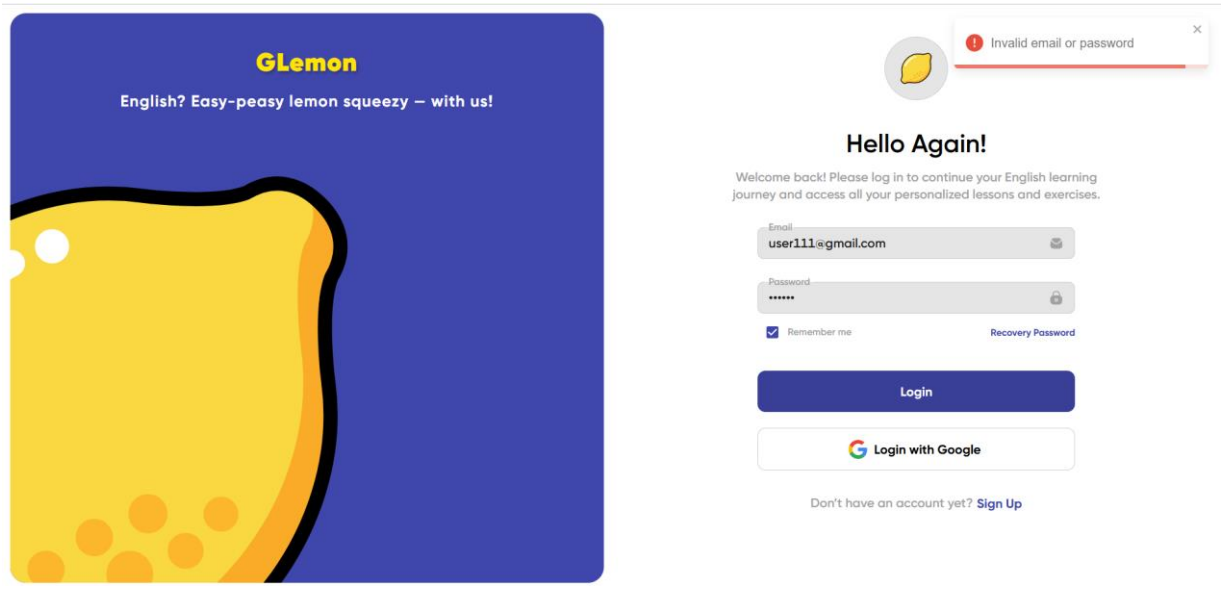


Рисунок 4.11 – Валідація форми аутентифікації при невірних даних

У разі якщо користувач ввів неправильні дані, ми отримаємо повідомлення про це (рис. 4.11).

Таким чином, можна пересвідчитися в тому, що аутентифікація користувача працює коректно.

4.2.3 Створення та тестування модуля

Потрбіно перевірити, чи працює функція створення нового навчального модуля.

Очікується, що після введення всіх необхідних даних (назва модуля, опис, картки) користувача має бути перенаправлено до сторінки модуля.

Також очікується, що при помилках, тобто якщо назва модуля порожня або є недостатньо карток, користувач має побачити відповідне повідомлення про помилку.

Перевіримо правильність створення модуля:

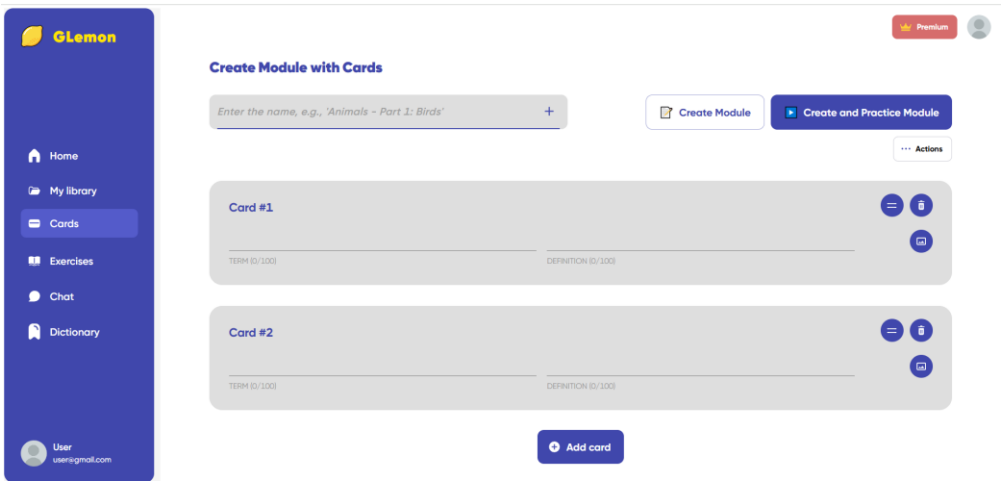


Рисунок 4.12 – Сторінка створення модуля

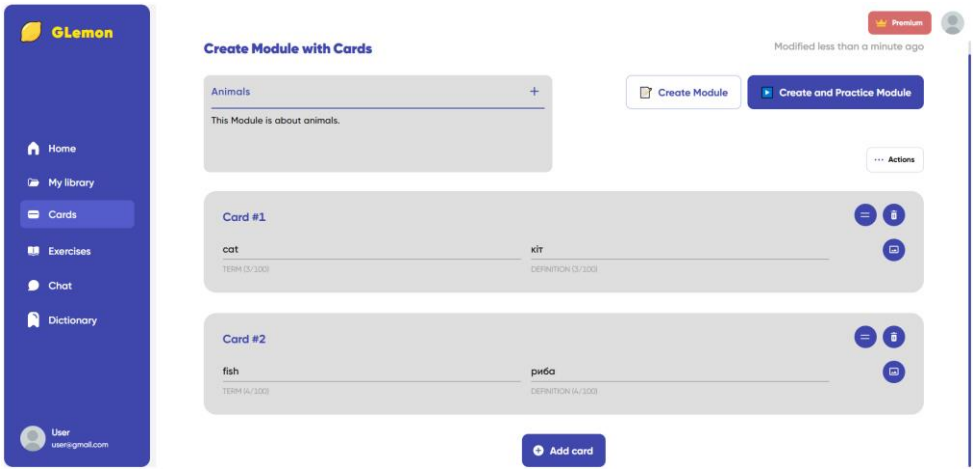


Рисунок 4.13 – Процес створення модуля

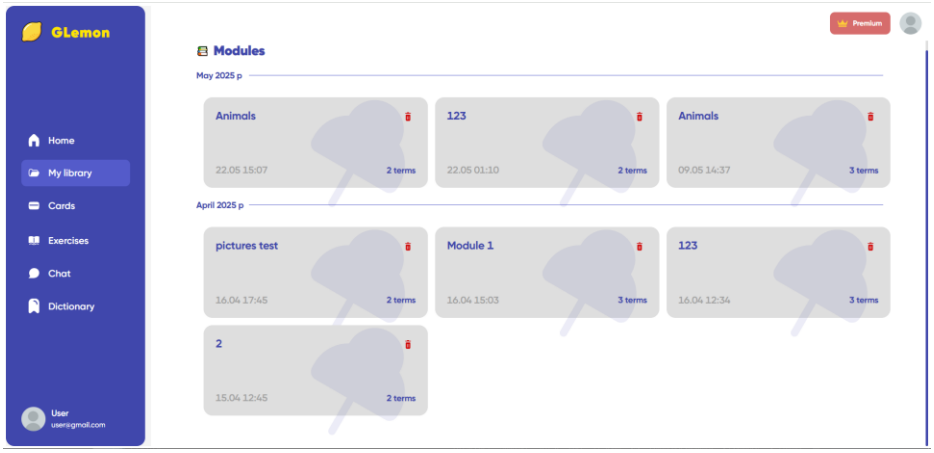


Рисунок 4.14 – Сторінка Library, де зберігаються усі створені модулі

Як бачимо, модуль Animals успішно створився. Він є серед списку в Бібліотеці модулів (рис. 4.14).

Подивимося, чи додався створений модуль у базі даних:

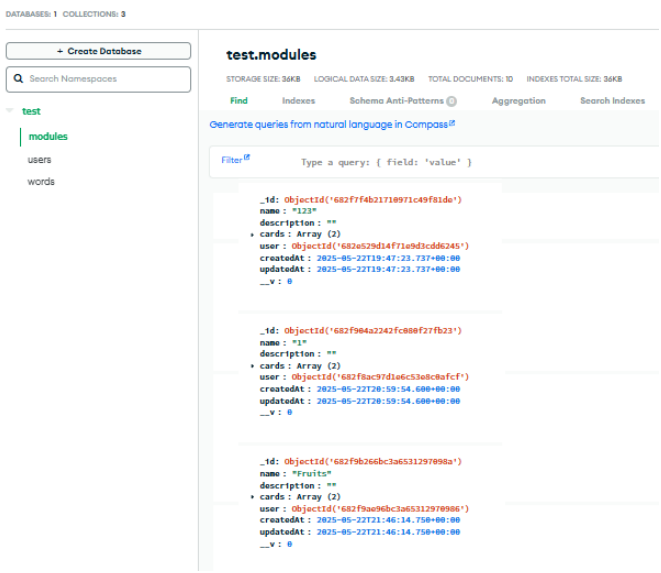


Рисунок 4.15 – Колекція Modules до створення модуля

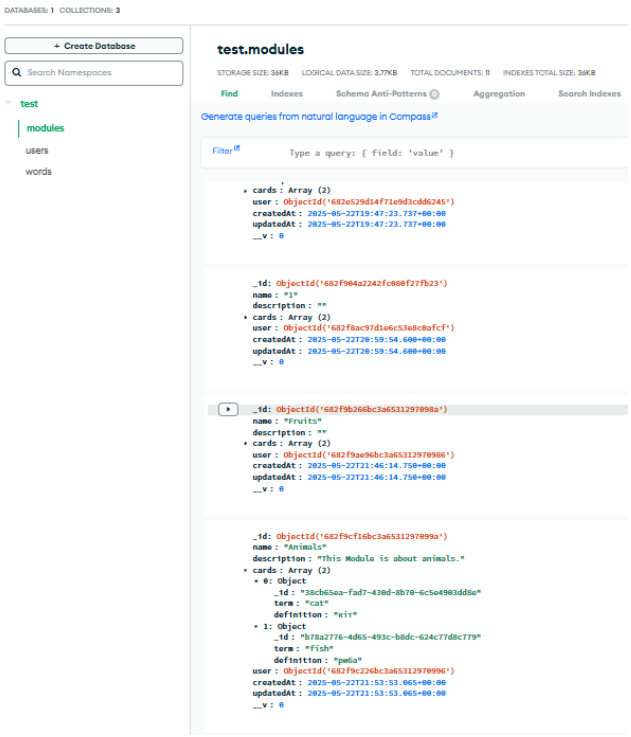


Рисунок 4.15 – Колекція Modules після створення модуля

Таким чином, на вищепоказаних скриншотах можна побачити, що при успішному створенні модуля, усі необхідні дані вносяться в базу даних у колекцію Modules і доступні до перегляду на стоірнці Library.

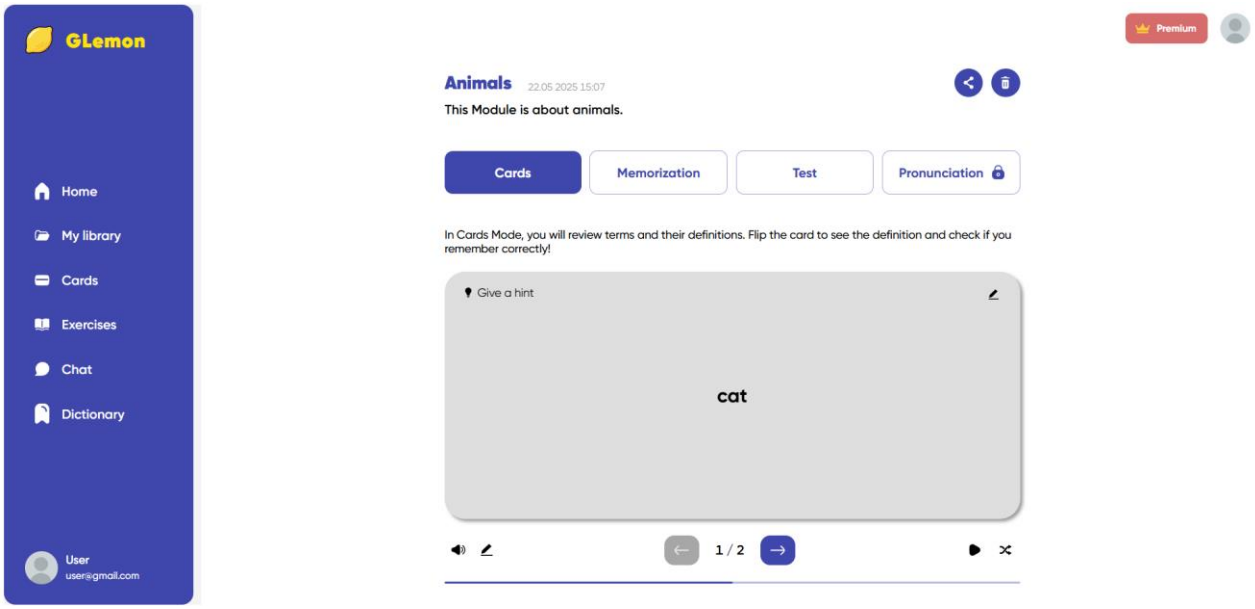


Рисунок 4.16 – Сторінка створеного модуля

Модуль з'явився в списку з правильним ім'ям, описом і кількістю карток (рис. 4.16).

Ця сторінка включає в себе різні компоненти та функції для взаємодії з модулями, такими як перегляд опису, зміну режиму навчання, а також видалення модуля. Модуль підтримує кілька режимів роботи, зокрема:

- Cards Mode — режим з картками (перегляд термінів та визначень).
- Memorization Mode — режим запам'ятовування термінів.
- Test Mode — режим тестування, в якому користувач має вибрати правильні визначення для термінів.
- Pronunciation Mode — режим вимови, де користувач може тренувати свою вимову за допомогою аудіо.

Користувач може перемикатися між цими режимами за допомогою кнопок, що змінюють вигляд активного режиму.

Розглянемо та перевіримо функціонал модуля в різних режимах ігор з картками для їх вивчення.

CardsMode (Режим карток). В цьому режимі користувач може переглядати терміни та їх визначення у вигляді карток. Кожна картка містить термін на одній стороні та визначення на іншій.

Картки можна перегортати для перегляду терміна або визначення. Є можливість прокручувати картки, переходячи до наступних або попередніх. Картки можуть містити додаткові зображення.

Ключовими функціями є перехід між терміном і визначенням картки, функції для перемикання між картками, можливість показати підказку для терміна або визначення, перемішування карток для зміни порядку їх відображення.

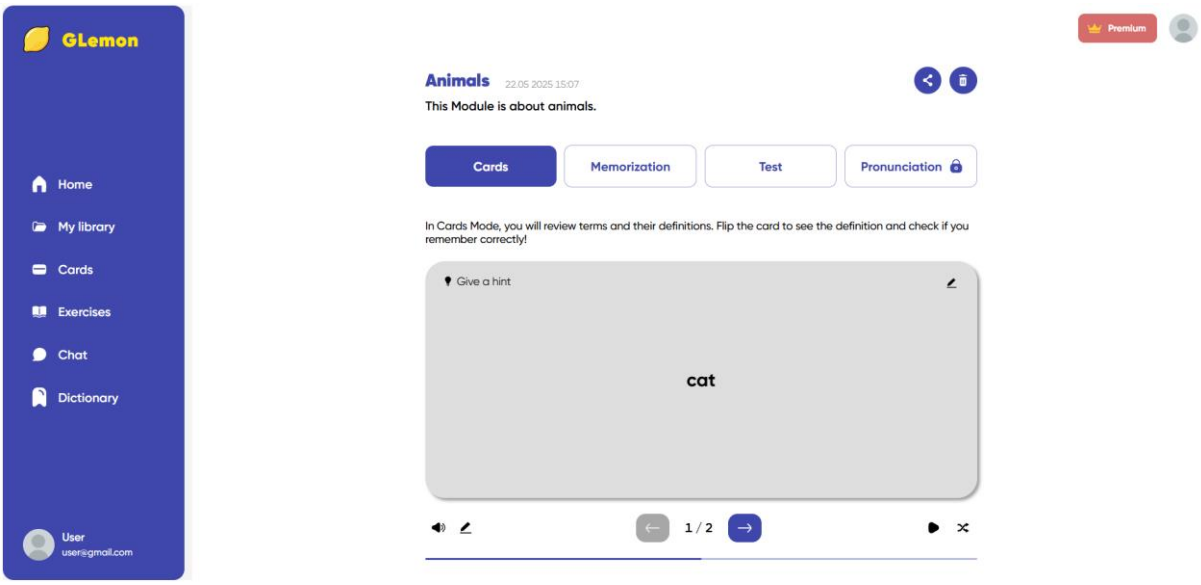


Рисунок 4.17 – Режим карток

Також є можливість редагувати картку. Давайте протестуємо:

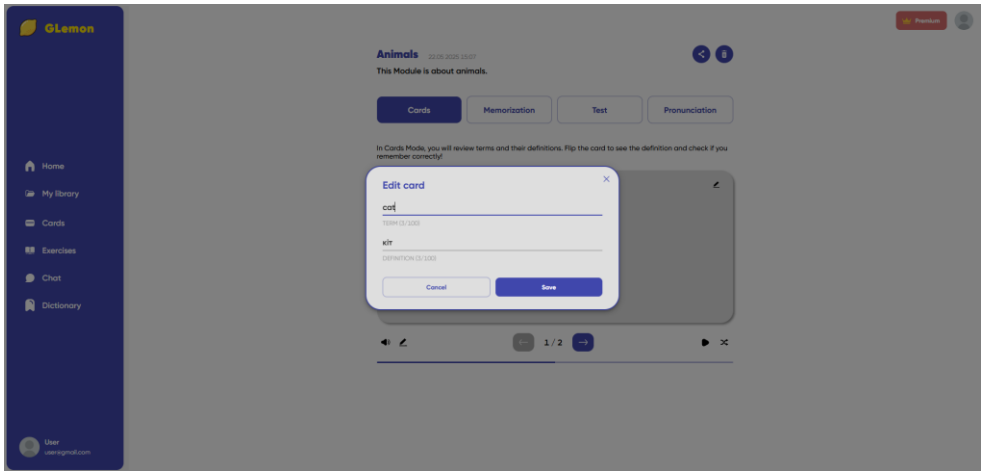


Рисунок 4.18 – Модальне вікно для редагування картки

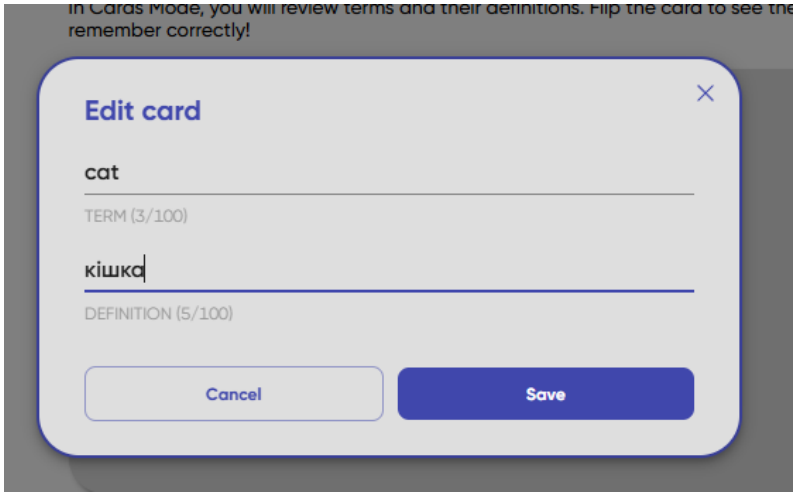


Рисунок 4.19 – Ввід нового визначення для картки

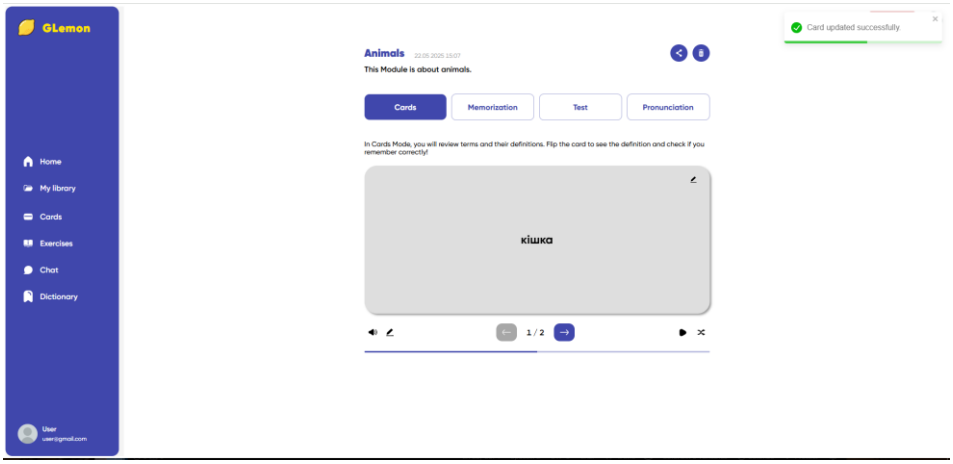


Рисунок 4.20 – Повідомлення про успішну зміну картки

MemorizationMode (Режим запам'ятовування). У цьому режимі користувач тренує своє запам'ятовування термінів. Користувач повинен ввести правильне визначення для терміна, що відображається на картці.

Користувач бачить термін та вводить відповідь-визначення в текстове поле. Після введення, система порівнює введене визначення з правильним варіантом та інформує користувача про правильність його відповіді. Після перевірки відповіді, користувач може перейти до наступної картки.

Ключовими функціями є обробка введеного користувачем тексту, відправка відповіді на перевірку, перехід до наступної картки після відповіді, відображення зворотного зв'язку після того, як користувач надіслав відповідь.

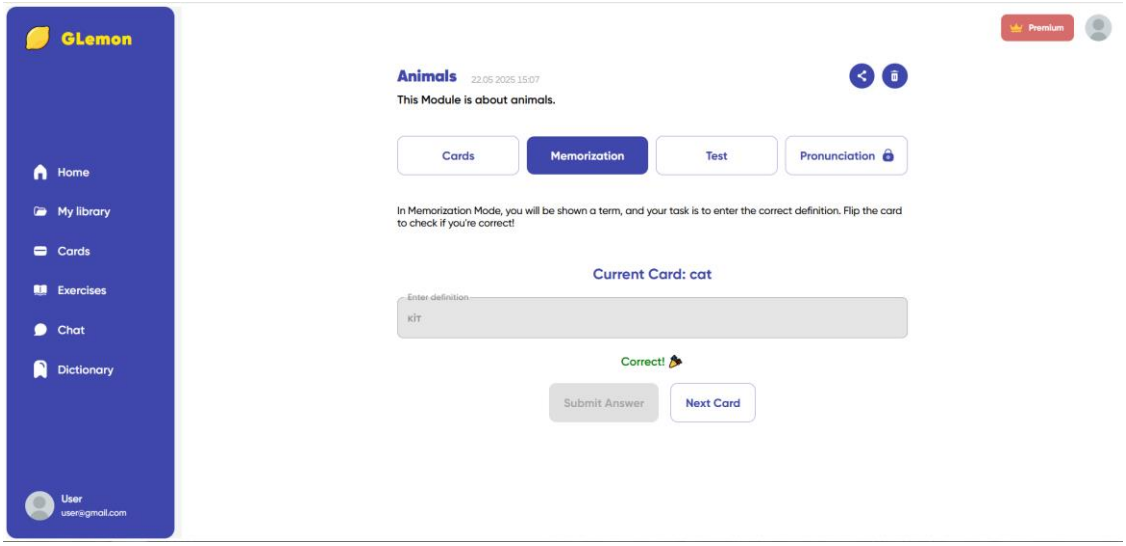


Рисунок 4.21 – Режим запам'ятовування

TestMode (Режим тесту). У цьому режимі користувач проходить тестування, у якому для нього генеруються звадання на основі наявних карток у модулі, у яких надається термін та потрібно вибрати правильне визначення з кількох варіантів.

Користувач вибирає варіант відповіді з кількох запропонованих. Після проходження всіх карток виводиться підсумковий результат (кількість правильних відповідей).

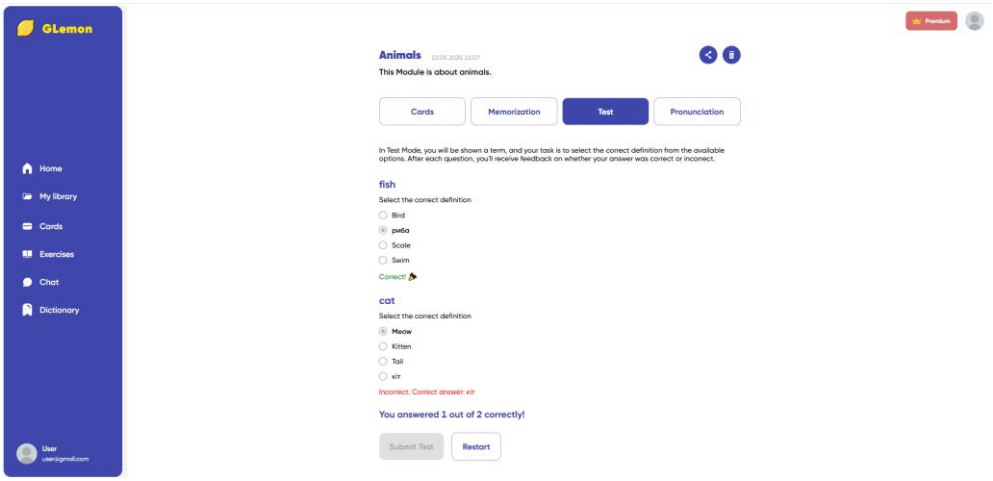


Рисунок 4.22 – Режим тесту

PronunciationMode (Режим вимови). У цьому режимі користувач тренує свою вимову, повторюючи за системою або записуючи власну вимову термінів.

Користувач може слухати терміни, озвучені системою (через SpeechSynthesis), і повторювати їх. Або ж користувач може записати свою вимову за допомогою мікрофона, і система перевірить, наскільки правильно він вимовив термін.

Ключовими функціями є відтворення терміну через синтезатор мови, перевірка вимови користувача, перехід до наступної картки після перевірки вимови.

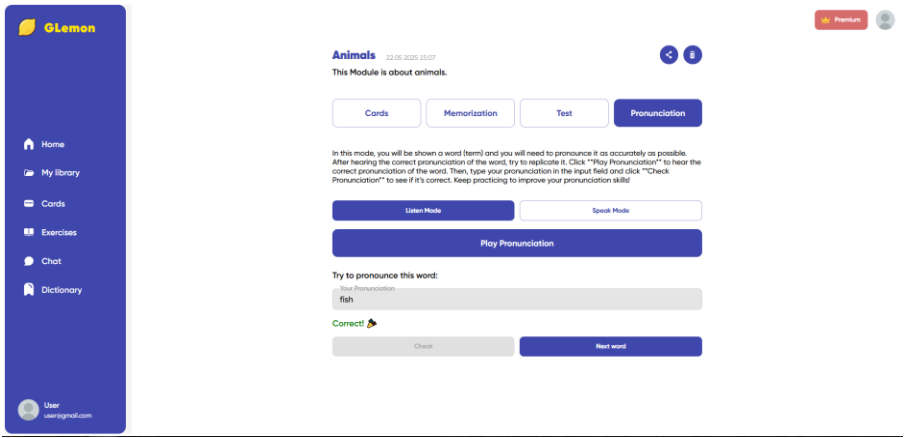


Рисунок 4.23 – Режим вимови

Отже, кожен режим має свою специфічну функціональність та забезпечує інтерактивний спосіб вивчення матеріалу, що дозволяє користувачам самим обирати найбільш зручний для них спосіб вивчення карток.

4.2.4 Гра Word Dictionary (Словник)

Гра Word Dictionary це по суті словник, який дозволяє користувачам зберігати вибрані слова у певному місці, отримувати більше інформації про слова та перевіряти їх переклад. Користувач в менжах цієї гри може додавати нові слова, редагувати їх, переглядати деталі та перевіряти правильність перекладів у вигляді гри. Для кожного слова відображаються додаткові дані, такі як транскрипція, синоніми та приклади використання, згенеровані ChatGPT. Крім того, передбачена функція видалення окремих слів або взагалі усіх слів із словника.

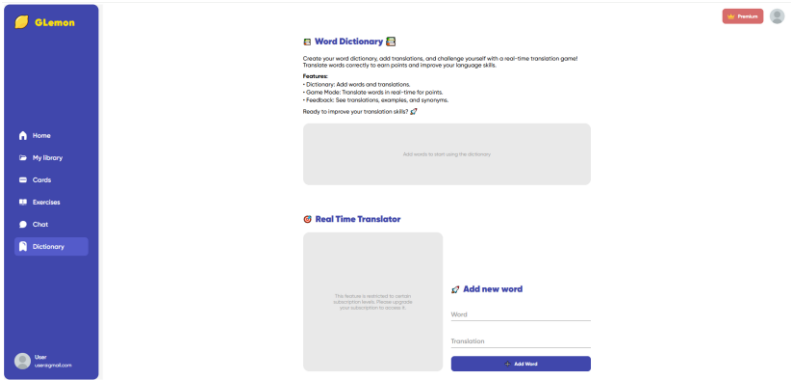


Рисунок 4.24 – Сторінка гри Word Dictionary

Протестуємо додавання нового слова до словника:

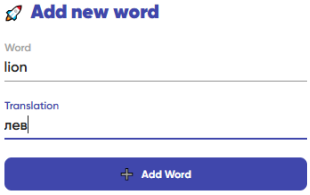


Рисунок 4.25 – Процес додавання слова до словника

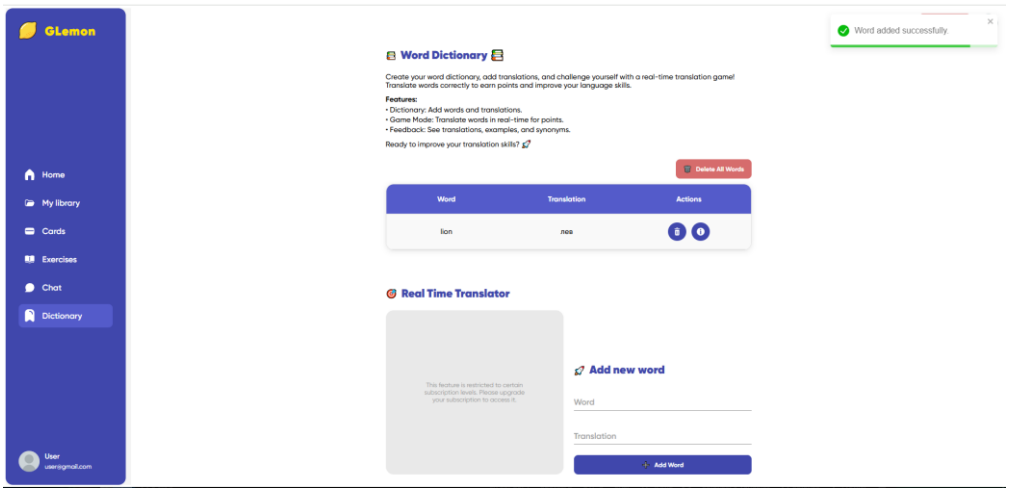


Рисунок 4.26 – Успішне додавання слова до словника

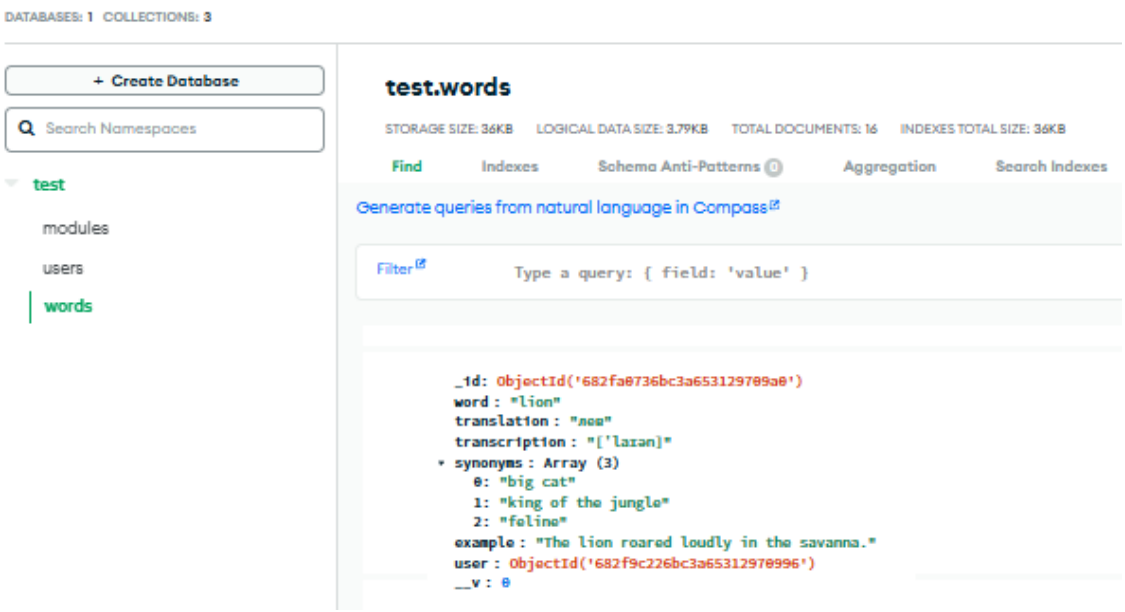


Рисунок 4.27 – Новостворене слово у базі даних

Як бачимо із рисунків 4.26 та 4.27 слово успішно створилося та додалося до словника, що на сторінці, що в базі даних.

Додатково, на сторінці є функція Real Time Translator, яка дозволяє користувачам перевіряти переклад слів у реальному часі. Вона надає можливість введення перекладу для слова, а система перевіряє правильність введеного перекладу та дає відповідний фідбек. Функція доступна тільки для користувачів із платною підпискою.

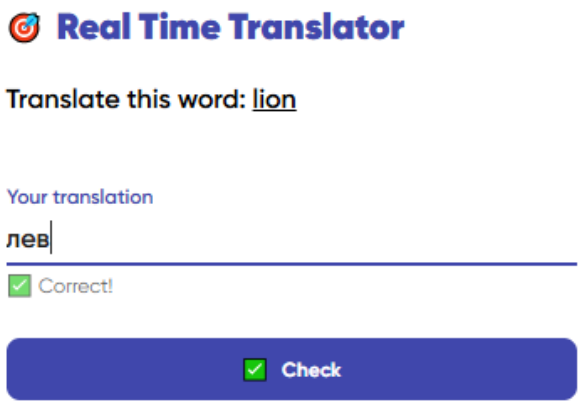


Рисунок 4.28 – Гра Real Time Translator

4.2.5 Гра Tongue Twister Challenge (Скоромовний челендж)

У грі Tongue Twister Challenge, у перекладі Скоромовний челендж, користувач має вимовити популярну скоромовку. Спочатку користувач має прочитати (є можливість прослухати скоромовку), а потім має записати голосовим свій варіант. Після завершення запису аудіофайл надсилається на сервер для транскрипції. Система використовує ChatGPT для оцінки вимови, видаючи результат у вигляді оцінки (від 0 до 100) і даючи поради з поліпшення вимови.

Отже, це інтерактивна гра, що поєднує елементи навчання вимови, аудіозапису та штучного інтелекту для зворотного зв'язку.

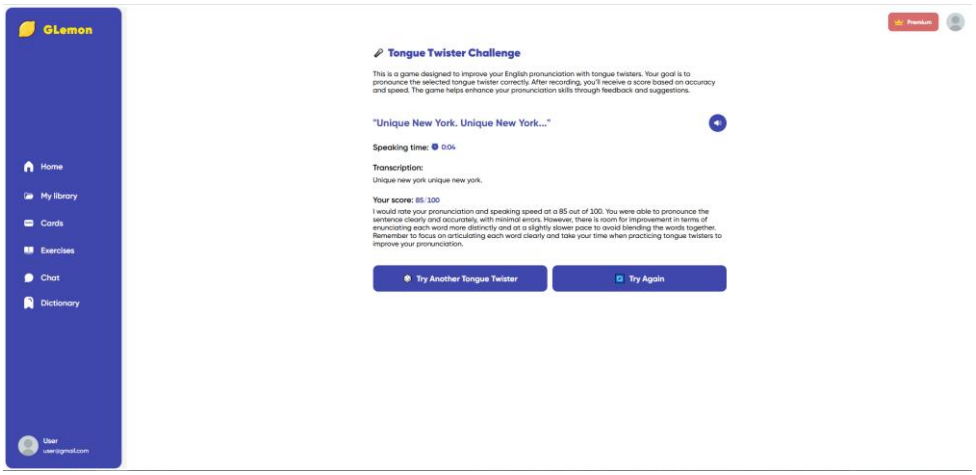


Рисунок 4.29 – Тестування гри Tongue Twister Challenge

4.2.6 Гра Vocabulary Builder (Розширення словникового запасу)

Гра Vocabulary Builder, у перекладі Розширення словникового запасу, створена для покращення словникового запасу, у якій користувач має заповнити пропуски в реченнях, використовуючи правильні слова з наданого списку. Гравець отримує перелік питань, де в кожному реченні є пропуск, який потрібно заповнити правильним словом. Вправи генеруються за допомогою OpenAI API, щоб створити унікальні завдання, які включають речення з пропусками і правильними відповідями.

Сторінка включає кілька етапів: створення вправ, введення відповідей, перевірка правильності відповідей та відображення підсумкового результату.

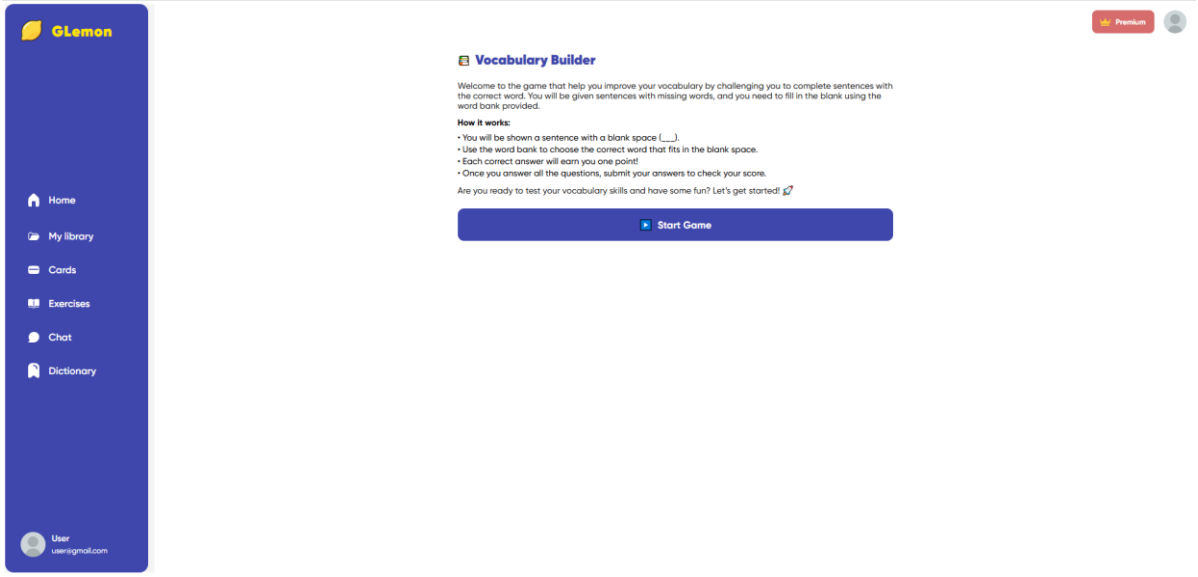


Рисунок 4.30 – Сторінка гри Vocabulary Builder

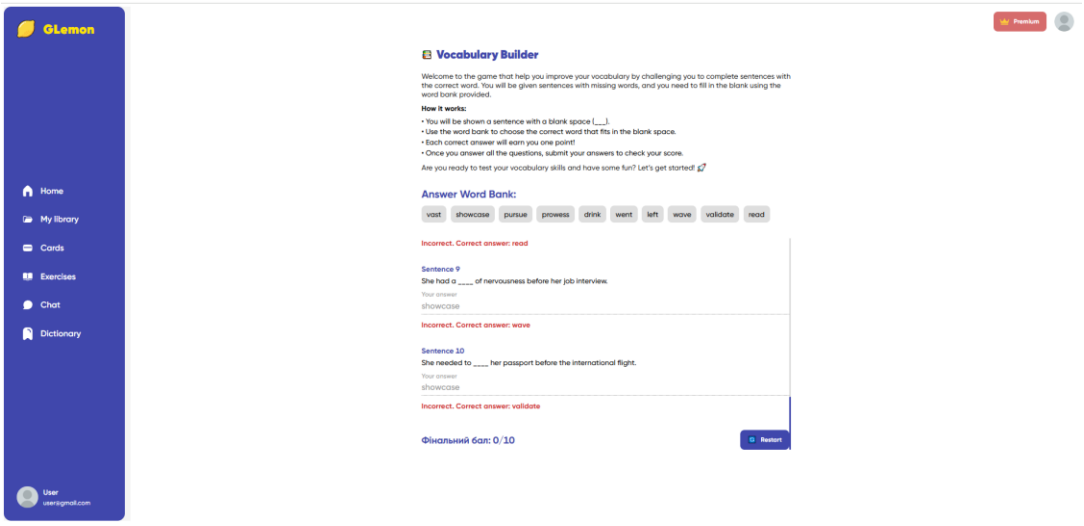


Рисунок 4.31 – Тестування гри Vocabulary Builder

4.2.7 Гра Word Explorer (Дослідник Слів)

Гра Word Explorer дозволяє користувачам розширювати свій словниковий запас за допомогою інтерактивних вправ. Тут генерується набір слів з перекладом, транскрипцією, синонімами та прикладами використання у вибраній зі списку категорії або за введеною користувачем категорією та дозволяє додавати ці слова до словника користувача.

Сторінка також дозволяє користувачу слухати вимову кожного слова, а також додає функціональність додавання нових слів до словника.

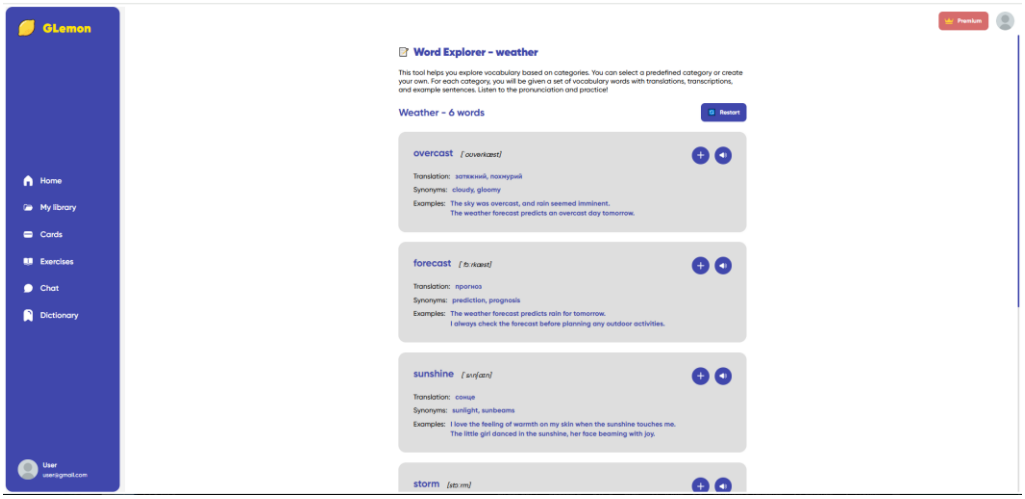


Рисунок 4.32 – Тестування гри Word Explorer

4.2.8 Гра Listening Practice (Практика слухання)

Гра Listening Practice, в перекладі Практика слухання, розроблена для розвитку навичок слухання англійської мови. Вона дозволяє користувачам вибрати рівень складності, акцент і швидкість мовлення та прослуховувати уривки текстів, які треба розпізнати для покращення розуміння на слух.

Перш ніж почати гру, користувач вибирає рівень складності (A1, A2, B1, B2, C1, C2), акцент (американський, британський чи австралійський) та швидкість мовлення (від 0.5x до 1.5x).

Після цього генерується вправа, що містить аудіо для прослуховування та питання, пов’язане з текстом.

Після прослуховування аудіо, користувач вводить свою відповідь в текстове поле і натискає кнопку для перевірки. Вправи генеруються за допомогою API OpenAI, а для перевірки відповідей користувача використовується алгоритм для аналізу семантики, щоб визначити, чи правильно він зрозумів головну ідею речення. Відповідь оцінюється як "правильна" чи "неправильна"і з відповідним зворотнім зв'язком.

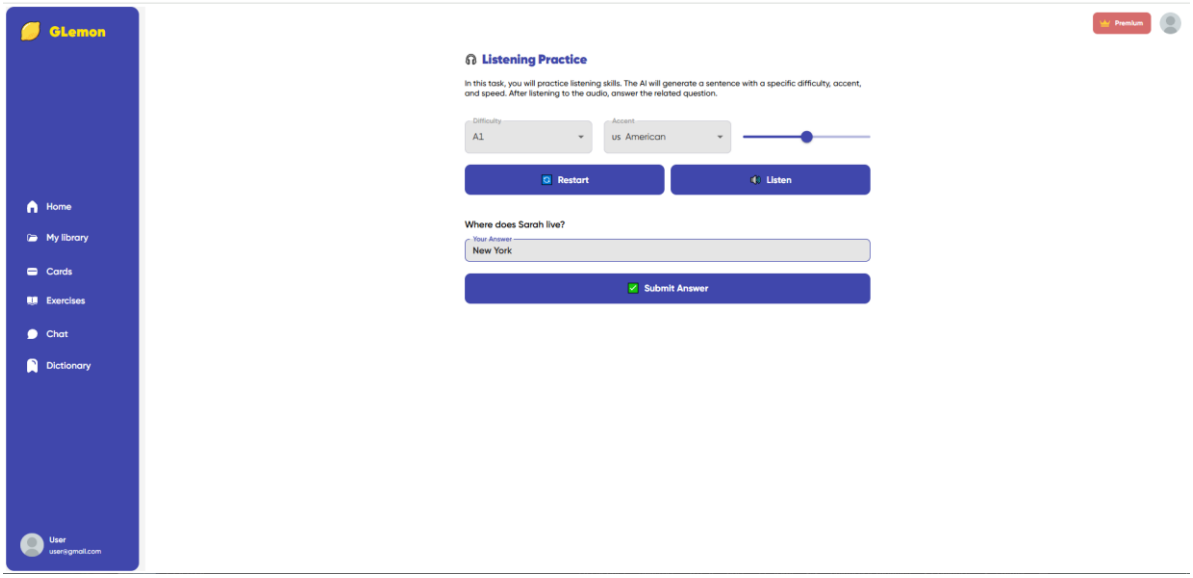


Рисунок 4.33 – Процес тестування гри Listening Practice

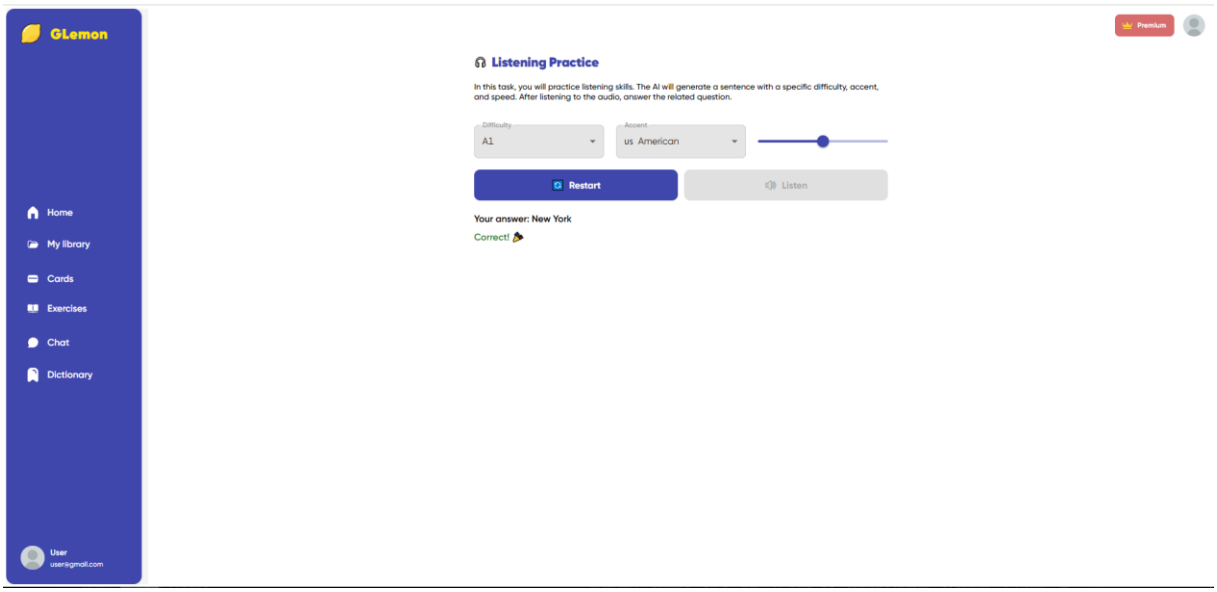


Рисунок 4.34 – Результат гри Listening Practice

4.2.9 Language Learning Exercises

Сторінка Exercises надає користувачам можливість вибирати категорії вправ, такі як граматика, лексика, слухання, або читання, та рівень складності для виконання інтерактивних вправ, що створюються за допомогою штучного інтелекту, згенерованого через API OpenAI, і можуть бути різних типів, включаючи завдання на вибір правильного варіанту, аудіо завдання для слухання та розуміння, а також тексти для читання з питаннями.

Після подачі відповідей користувач отримує зворотний зв'язок про правильність відповідей разом із поясненнями, які доступні, якщо у користувача куплена підписка.

Давайте протестуємо пару категорій, наприклад Grammar для рівня B2 та Listening для рівня C1:

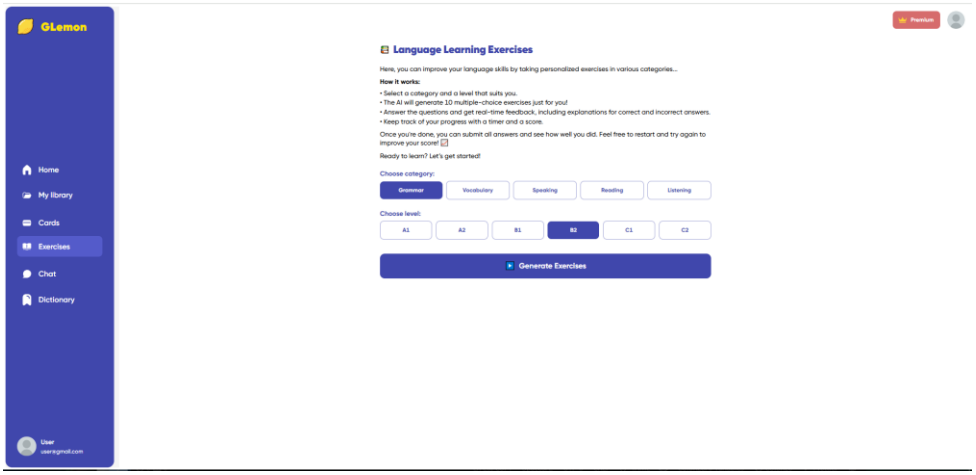


Рисунок 4.35 – Налаштування на генерацію завдань Grammar для рівня B2

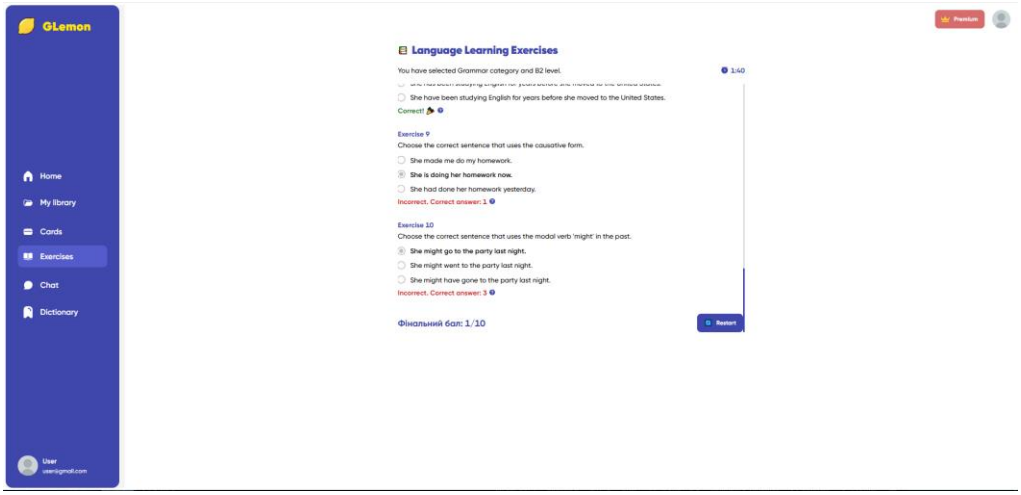


Рисунок 4.36 – Результати тестування Grammar для рівня B2

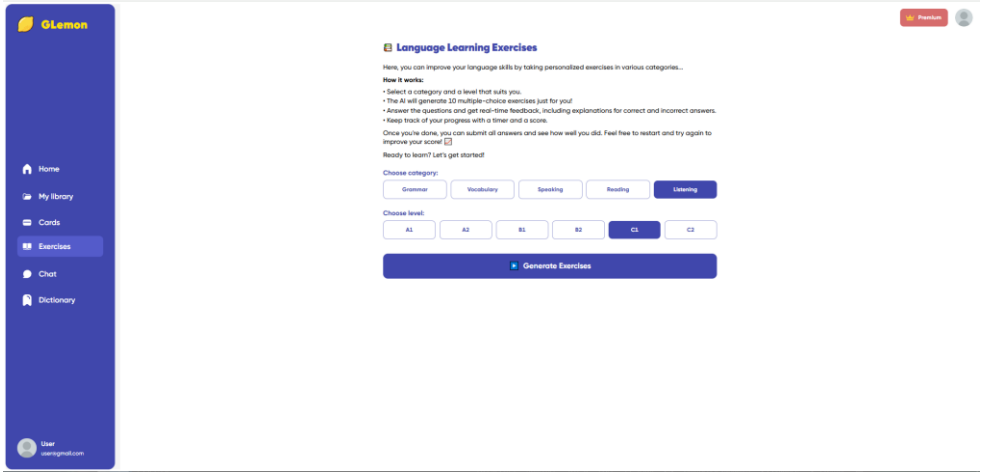


Рисунок 4.37 – Налаштування на генерацію завдань Listening для рівня C1

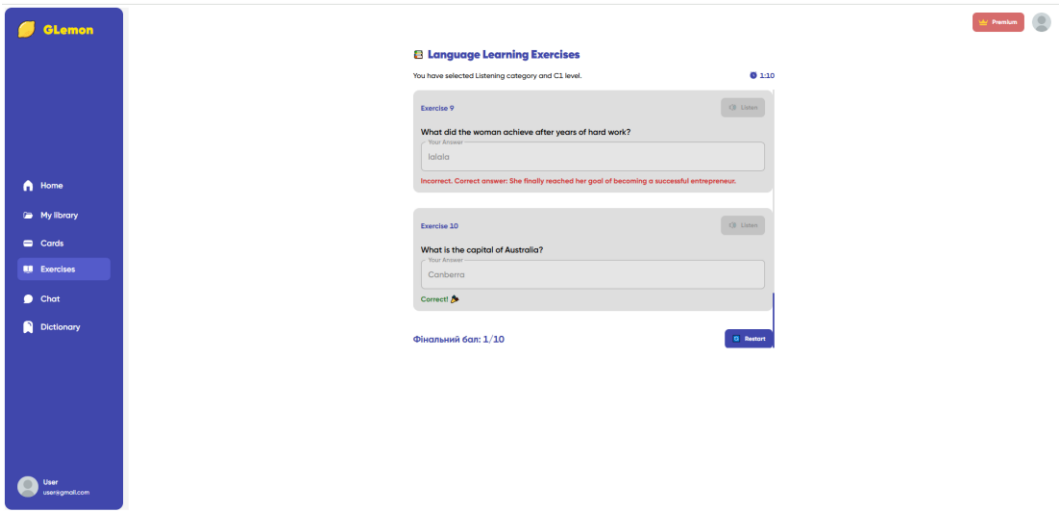


Рисунок 4.38 – Результати тестування Listening для рівня C1

4.2.10 Гра AI Speaking Buddy (AI Співрозмовник)

Гра AI Speaking Buddy, в перекладі AI Співрозмовник, створена для того, щоб користувачі мали можливість практикувати англійську мову через розмови з віртуальним асистентом. Користувач може вибрати один із запропонованих сценаріїв (наприклад, розмова в кафе, на аеропорту, під час роботи або замовлення їжі), і асистент почне діалог, генеруючи відповіді в реальному часі на основі вибраної теми та допомагаючи удосконалювати мовні навички.

Користувач може записати своє питання або відповідь, або записати голос, а система згенерує коректну відповідь для коректного продовження діалогу.

Система використовує Vosk для розпізнавання голосу в реальному часі, перетворюючи аудіозаписи користувача на текст. Коли користувач завершив запис, транскрибований текст передається в систему для подальшої обробки. Для аналізу текстових запитів та взаємодії з користувачем використовується Rasa, яка визначає наміри користувача та вибирає відповідний сценарій діалогу. У разі відсутності відповіді від Rasa, відбувається генераці природної відповіді від OpenAI, базуючись на попередніх повідомленнях.

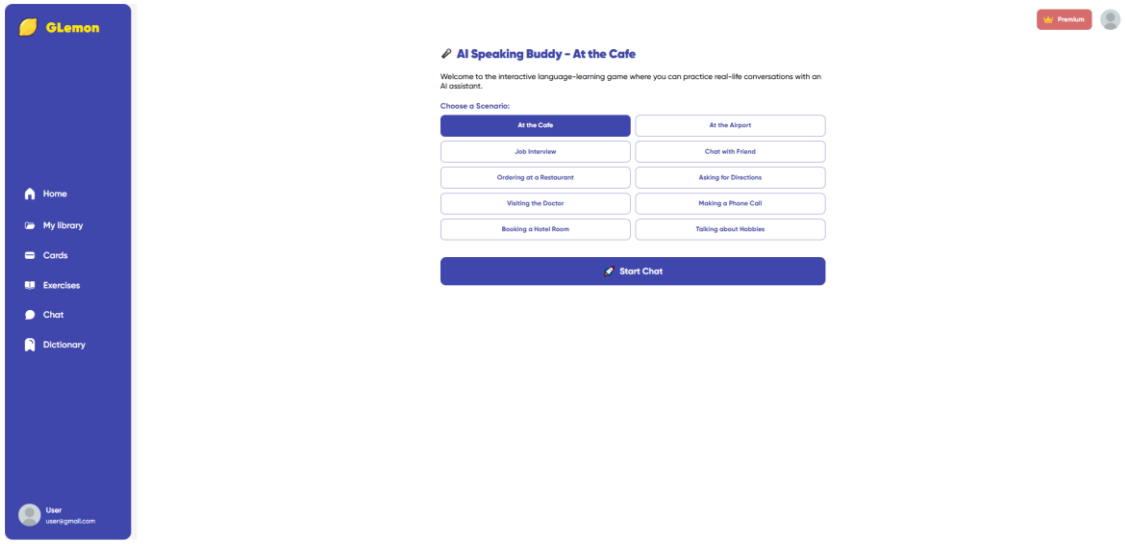


Рисунок 4.39 – Сторінка гри AI Speaking Buddy

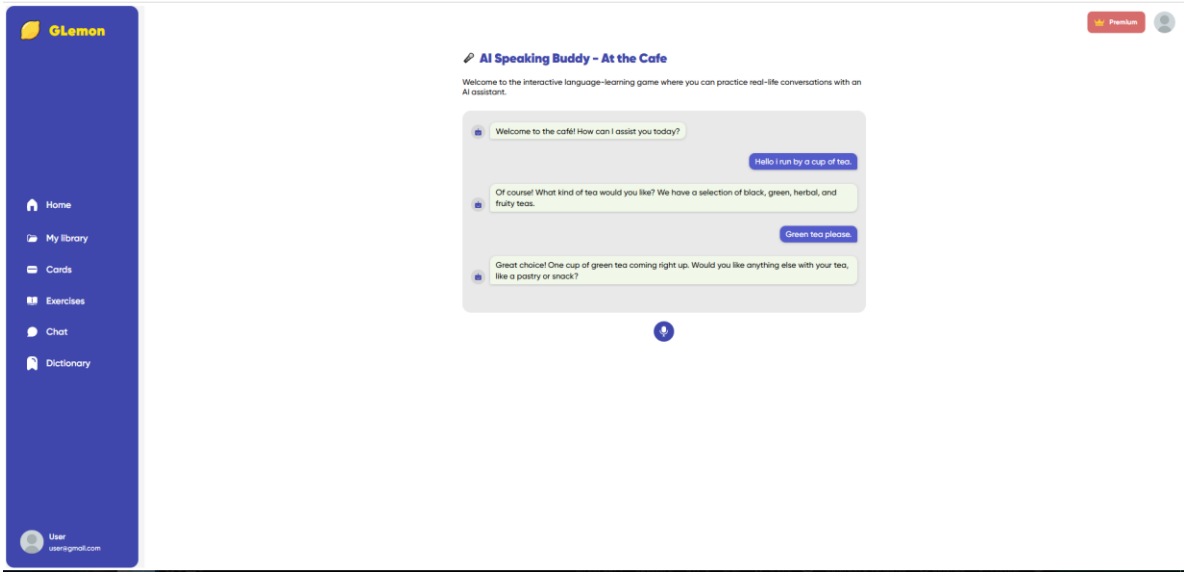


Рисунок 4.40 – Процес спілкування із асистентом

4.2.11 Chat

Компонент Chat забезпечує інтерактивний чат, в якому користувач може спілкуватися з ботом як через текстові, так і через голосові повідомлення. Після відправки повідомлення, воно надсилається до Rasa для обробки. Якщо відповідь від Rasa є заблокованою або недостатньо інформативною, система виконує fallback і звертається до OpenAI для отримання корисної відповіді.

Усі повідомлення, як користувача, так і бота, зберігаються в стані та відображаються користувачу в чаті.

Користувач також може використовувати голосову команду для запису повідомлення через мікрофон, при цьому система автоматично розпізнає та транскрибує голосовий запис за допомогою Vosk, після чого текст проходить вищеописану обробку для генерації відповіді, яка буде відтворена голосом через Speech Synthesis API.

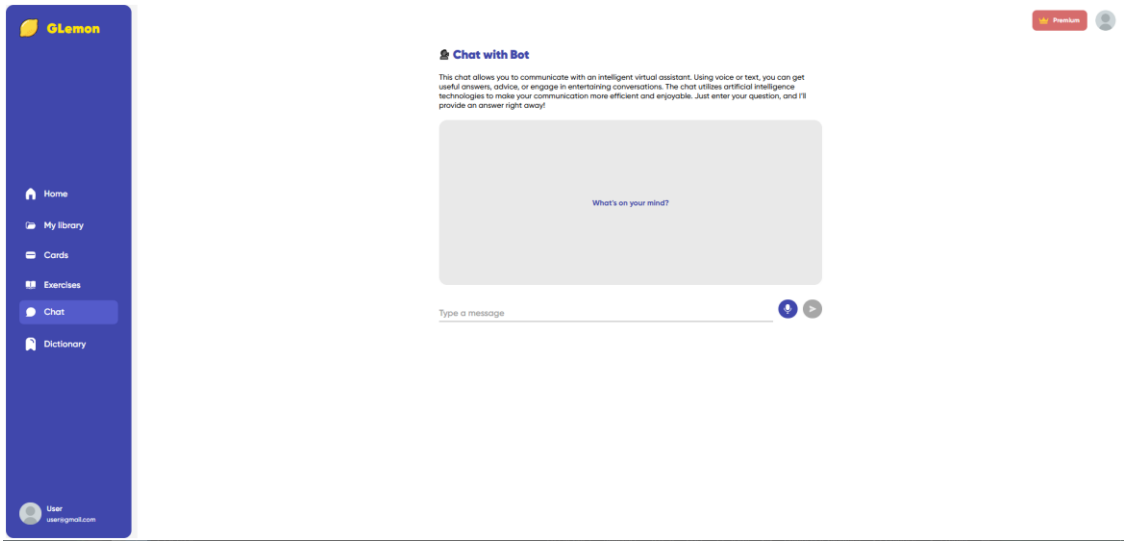


Рисунок 4.40 – Сторінка Chat

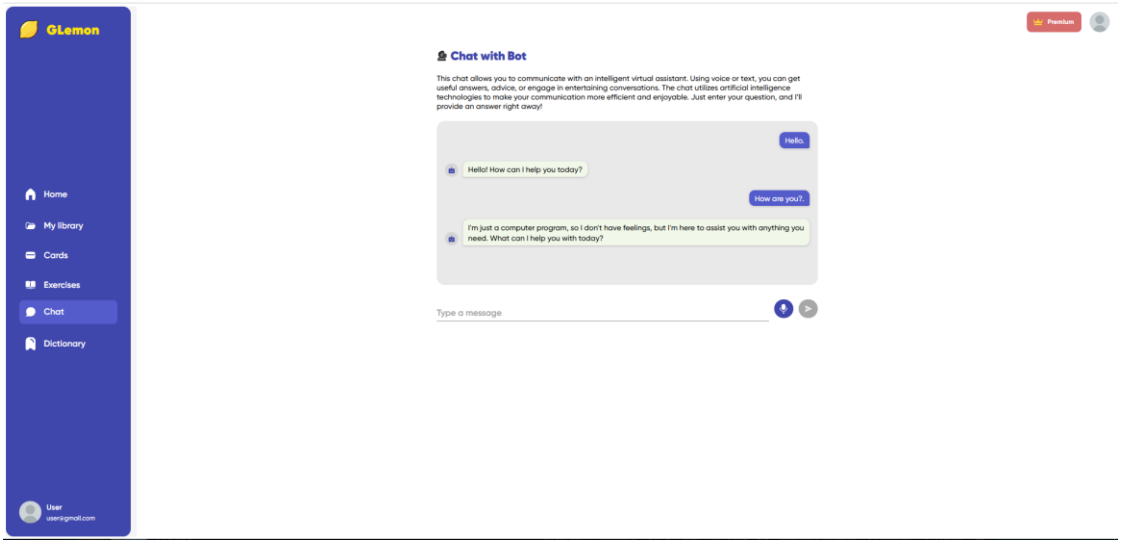


Рисунок 4.41 – Процес спілкування з ботом

4.3 Інтеграційне тестування

Інтеграційні тести у цьому вебдодатку мають на меті перевірити коректність взаємодії між різними компонентами: клієнтською та серверною частинами, API, базою даних і зовнішніми сервісами (OpenAI, Rasa, Vosk).

4.3.1 Тест завантаження, відображення, додавання, видалення слів

Перевіримо, що при зверненні до API отримуються правильні дані і вони коректно відображаються на сторінці. Кроки для тестування:

- Відкрити сторінку словника.
- Викликати API для отримання слів.
- Перевірити, що дані з API відповідають очікуваним.
- Переконатися, що слова відображені у таблиці.

Очікується, що слова коректно завантажуються і відображаються без помилок.

Перевіримо це:

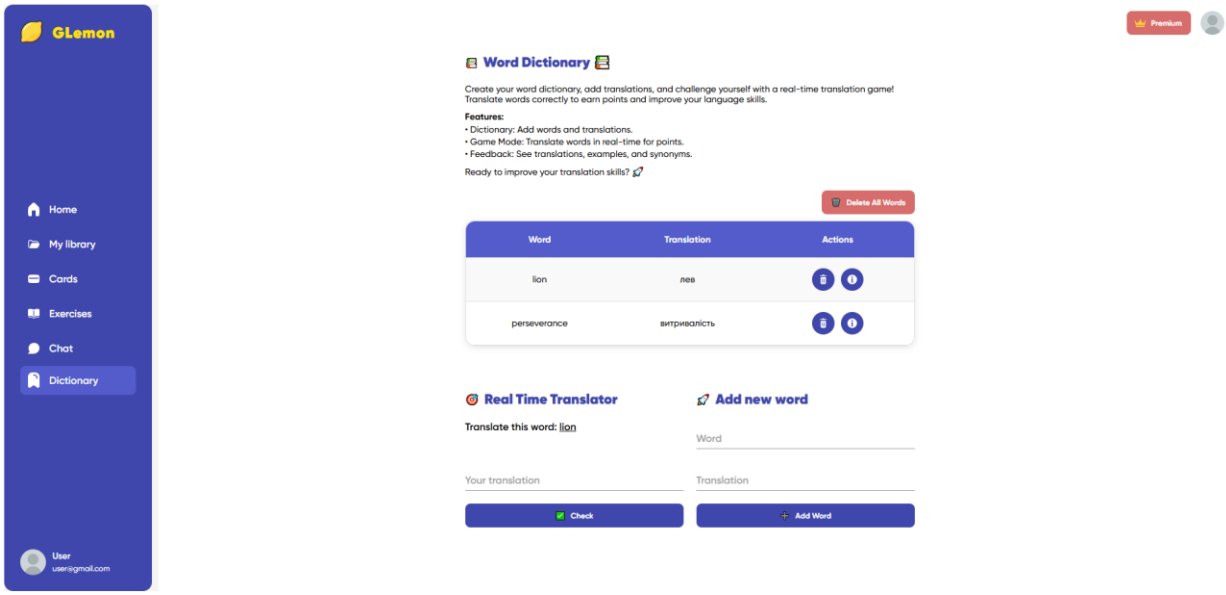


Рисунок 4.42 – Сторінка словника

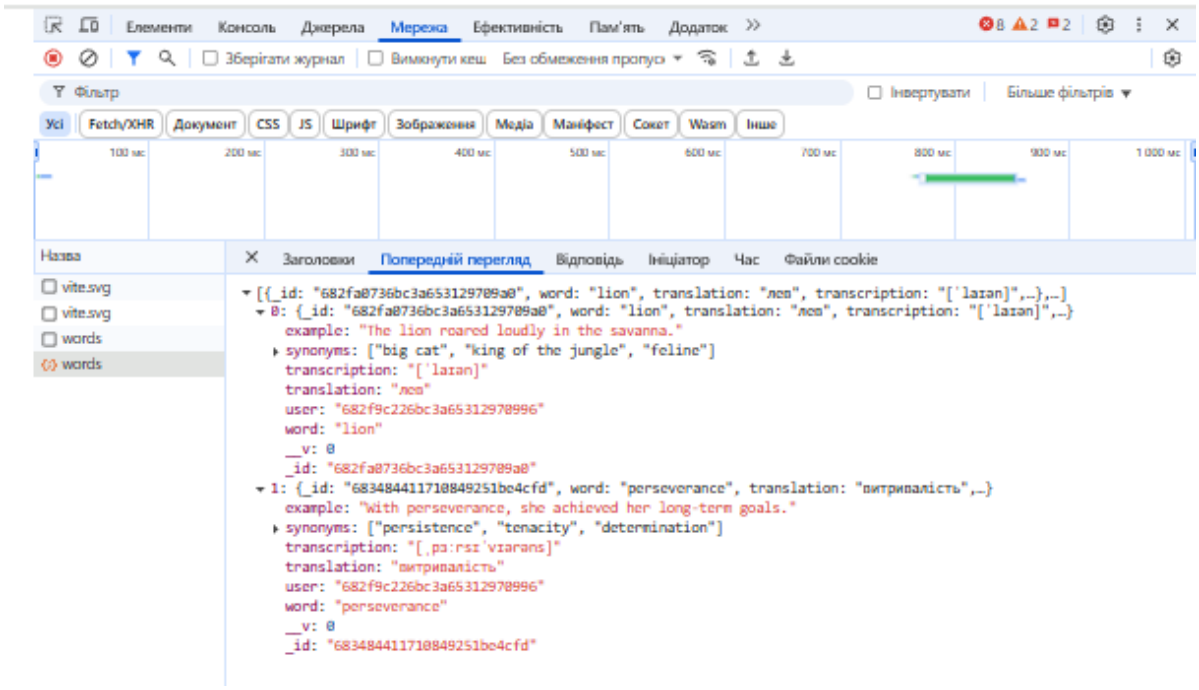


Рисунок 4.43 – Запит на отримання слів в консолі розробника

Отже, як видно із рисунків 4.42, 4.43, отримані дані співпадають із відображеними словами, тобто інтеграція між фронтендом, бекендом та БД працює добре.

Далі перевіримо, що нове слово можна додати прямо на стоірнці і воно потрапляє у базу даних.

Кроки для тестування:

- Відкрити форму додавання слова.
- Заповнити поля та відправити запит.
- Перевірити відповідь API (успіх, статус 201).
- Перевірити, що слово відобразилося у списку.
- Перевірити базу, що слово реально збереглося.

Очікується, що слово успішно додається і з’являється в інтерфейсі.

Перевіримо це:

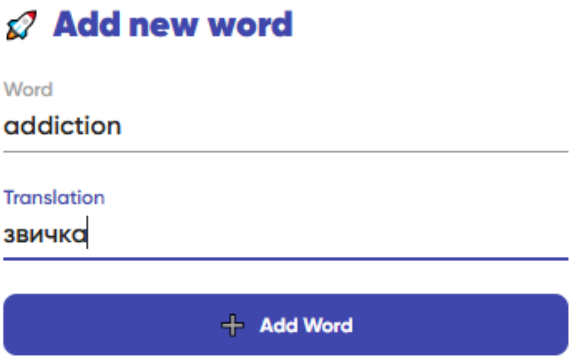


Рисунок 4.44 – Форма додавання слова

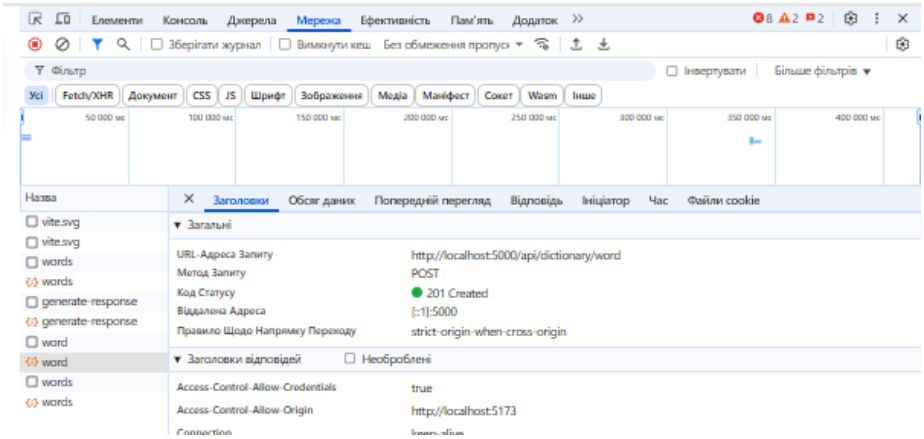


Рисунок 4.45 – Відповідь API (успіх, статус 201)

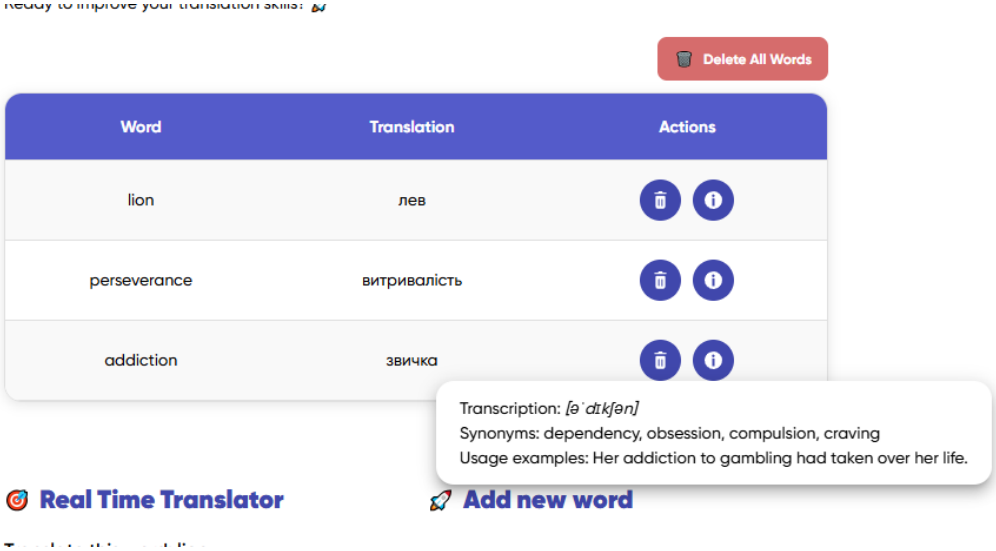


Рисунок 4.46 – Відображення новоствореного слова у списку

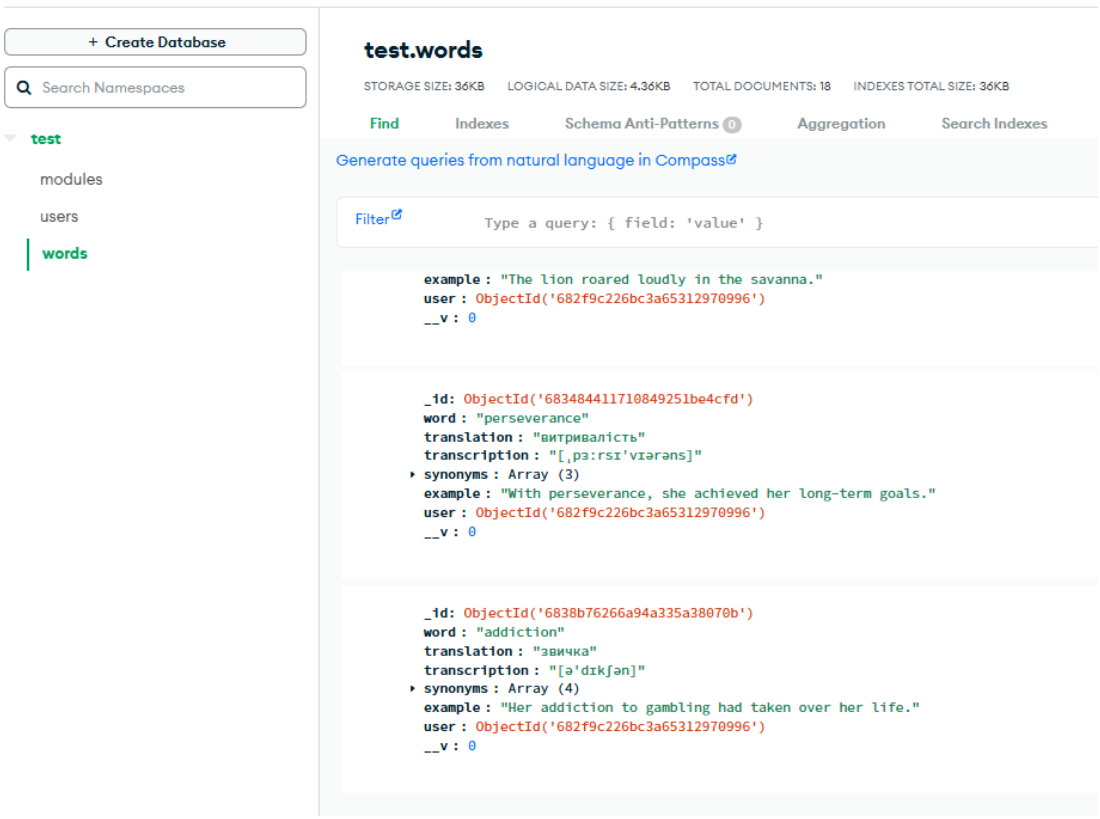


Рисунок 4.47 – Наявність новоствореного слова у БД

Отже, із рисунків 4.44, 4.45, 4.46 та 4.47 можна зробити висносок, що додавання слова працює, дані зберігаються коректно.

Далі перевіримо, що видалення слова працює і це видно на сторінці та в базі даних.

Кроки для тестування:

- Вибрати слово у списку.
- Натиснути кнопку "видалити".
- Перевірити відповідь API (статус 200).
- Переконатися, що слово більше не відображається.
- Перевірити БД — слова немає.

Очікується, що слово успішно видаляється із системи.

Перевіримо це. Оберемо слово «addiction» та натиснемо кнопку видалення:

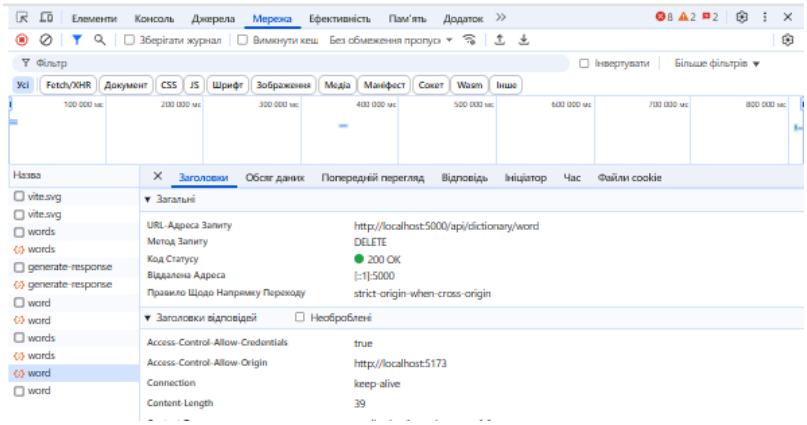


Рисунок 4.48 – Відповідь API (успіх, статус 200)

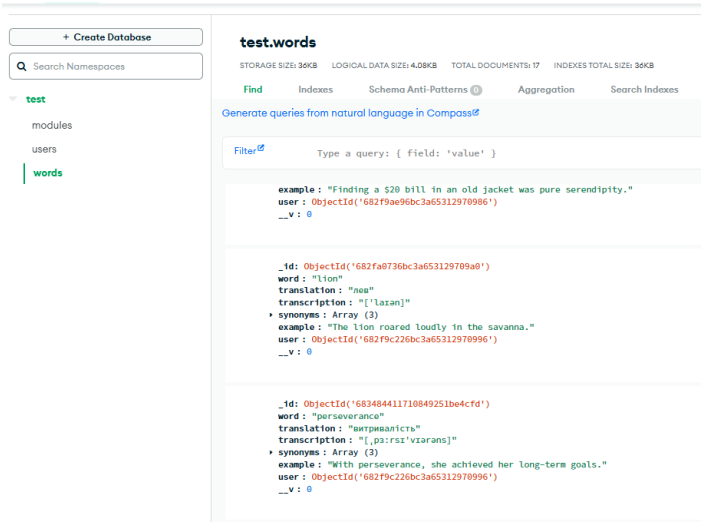


Рисунок 4.49 – Слово в БД видалилось

Отже, із рисунків 4.48, 4.49, можна зробити висносок, що видалення працює коректно.

4.3.2 Тест інтеграції з OpenAI

Перевіримо, що фронтенд коректно отримує відповіді від OpenAI (наприклад, для генерації вправ чи оцінювання).

Кроки для тестування:

- Запустити сценарій, який відправляє запит до OpenAI.
- Перевірити, що відповідь приходить.

– Перевірити коректність відображення результату.

Очікується, що відповідь AI отримується і відображається.

Перевіримо це. Оберемо гру «Language Learning Exercises» для категорії Vocabulary рівня B2:

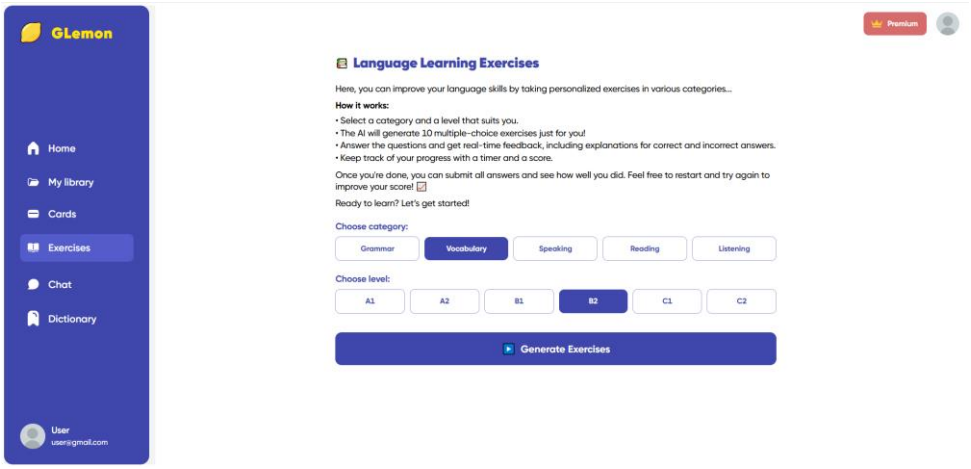


Рисунок 4.50 – Вибір опцій для генерації вправ

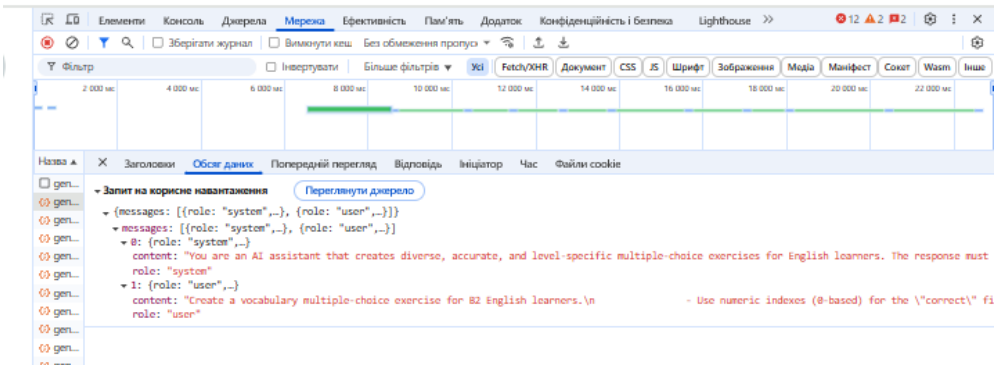


Рисунок 4.51 – Сценарій, який відправляє запит до OpenAI

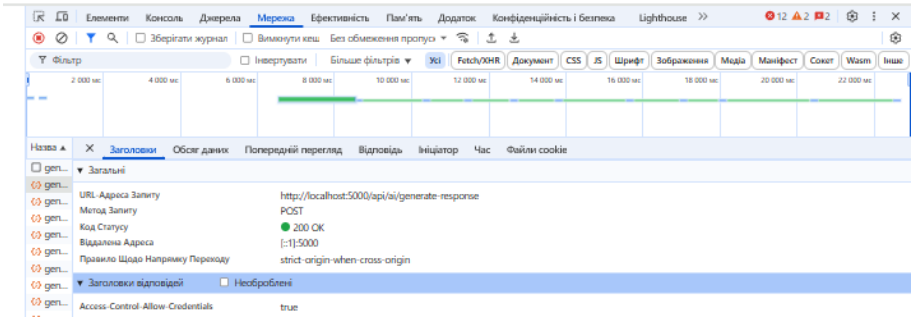


Рисунок 4.52 – Відповідь API (успіх, статус 200)

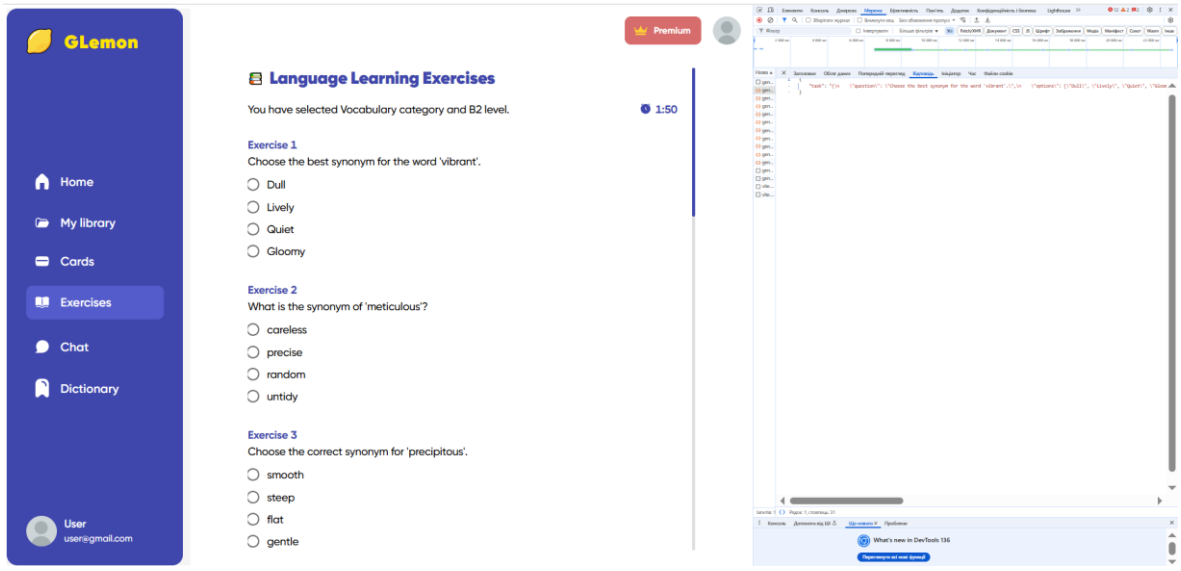


Рисунок 4.53 – Отримані дані із OpenAI

Отже, із рисунків 4.50, 4.51, 4.52 та 4.53 можна зробити висносок, що інтеграція з OpenAI працює коректно.

4.3.3 Тест інтеграції з Rasa

Перевіримо коректність взаємодії з AI-асистентом Rasa.

Кроки для тестування:

- Відправити повідомлення користувача через чат.
- Перевірити відповідь від Rasa.
- Переконатися, що fallback працює (перехід на OpenAI якщо Rasa не відповідає).

Очікується, що AI-асистент коректно відповідає або делегує.

Перевіримо це та перейдемо на сторінку «Chat». Для перевірки роботи Rasa на відомі інтенти було надіслано повідомлення: “hello” — очікувалося розпізнавання інтенту greet і відповідь: “Hello! How can I help you today?”.

Нижче наведені скріншоти отриманих відповідей від OpenAI (через fallback або інтерфейс інтеграції), що підтверджують правильність роботи системи.

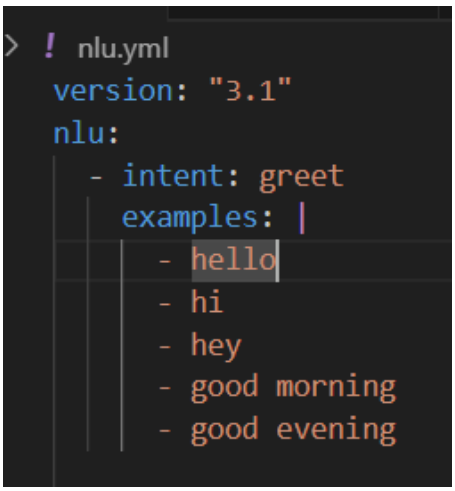


Рисунок 4.54 – Визначення інтенту greet

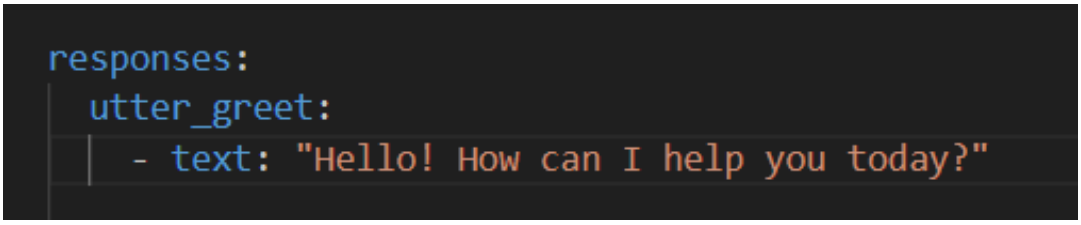


Рисунок 4.55 – Очікувана відповідь для інтенту greet

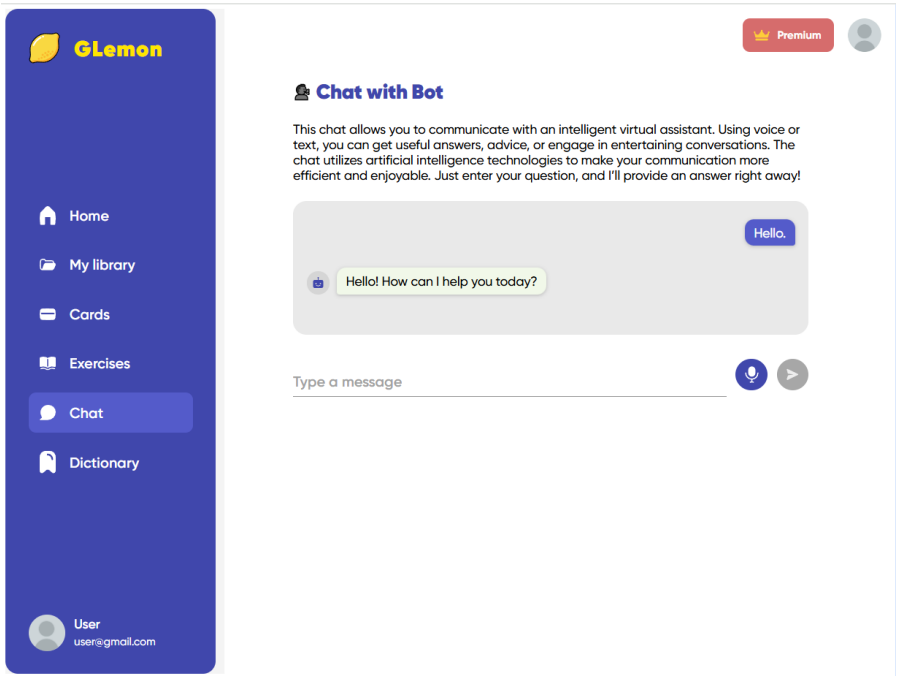


Рисунок 4.56 – Коректна відповідь бота у чаті після розпізнавання інтенту greet

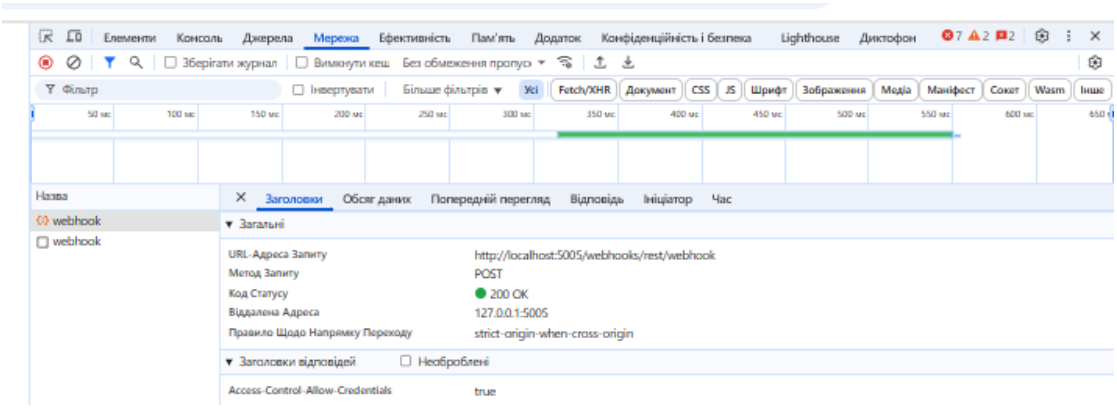


Рисунок 4.57 – Відповідь від серверу Rasa (успіх, статус 200)

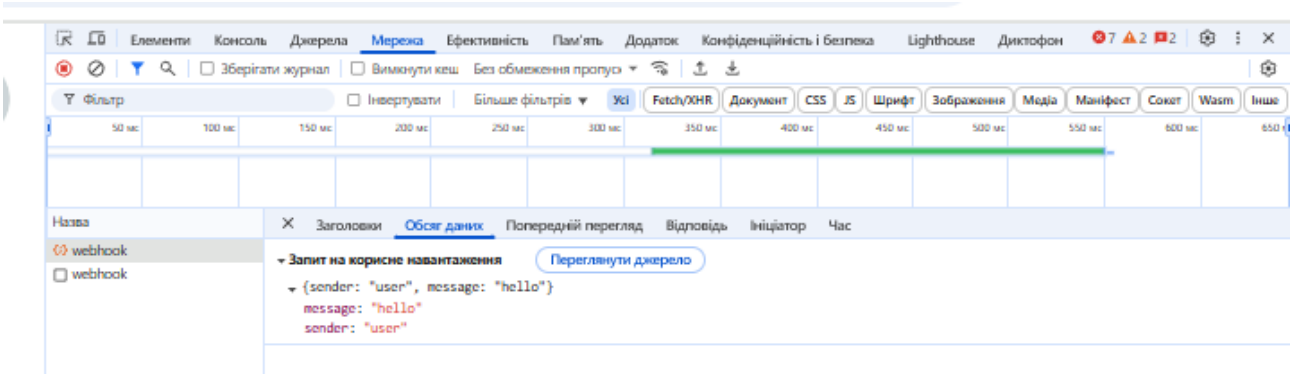


Рисунок 4.58 – Дані для відправки до серверу Rasa

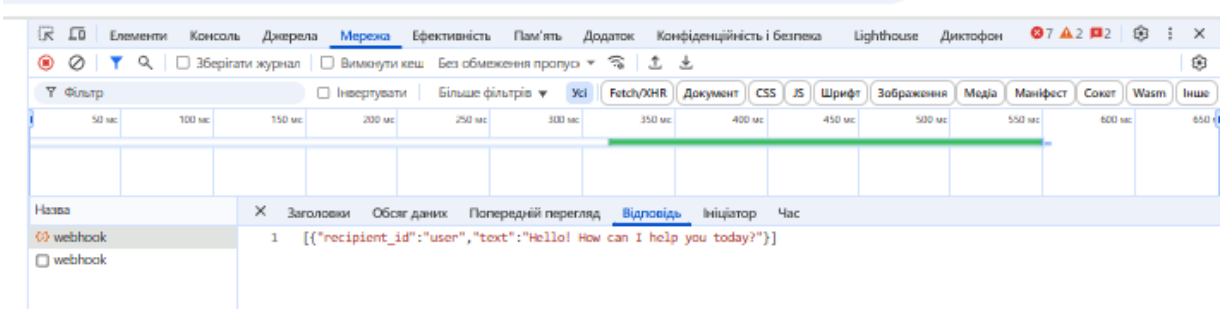


Рисунок 4.59 – Отримані дані із серверу Rasa

Для перевірки fallback-механізму було надіслано повідомлення: “tell me a joke” — повідомлення, на яке Rasa не має відповіді у тренувальних даних.

Очікувалося, що в такому випадку відбудеться делегування на OpenAI для генерації відповіді.

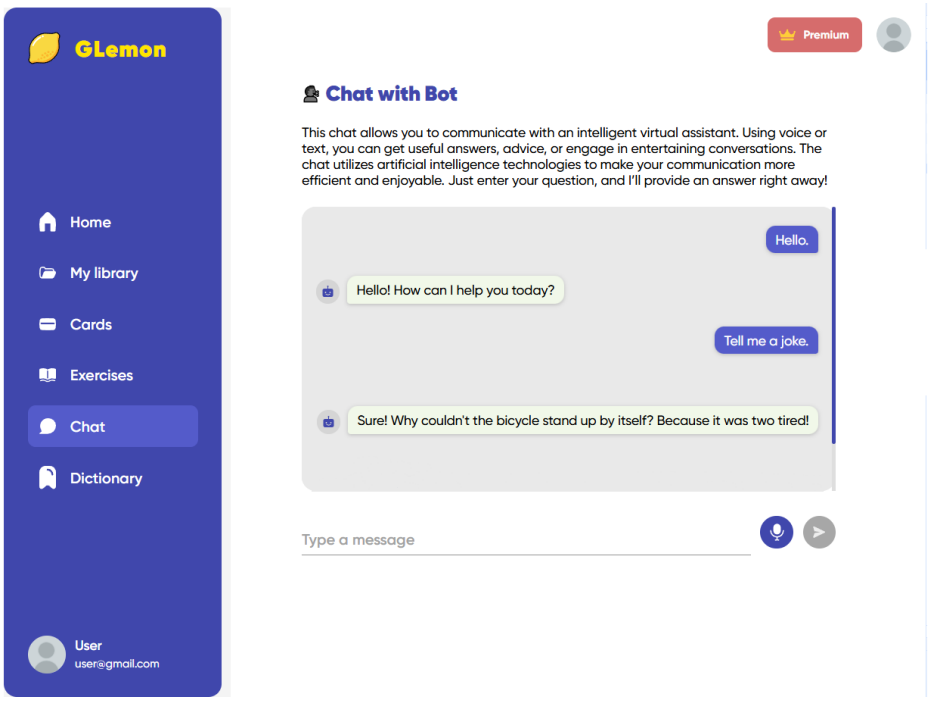


Рисунок 4.60 – Чат із отриманою відповіддю від OpenAI

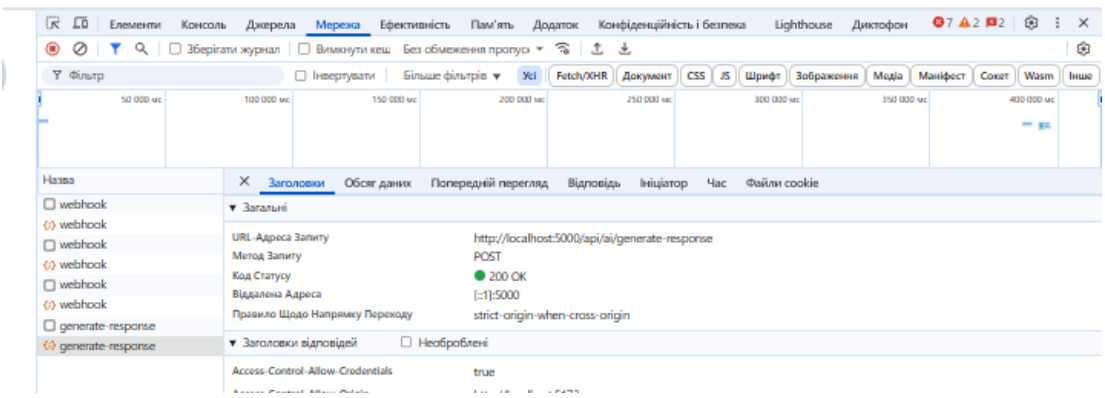


Рисунок 4.61 – Відповідь API (успіх, статус 200)

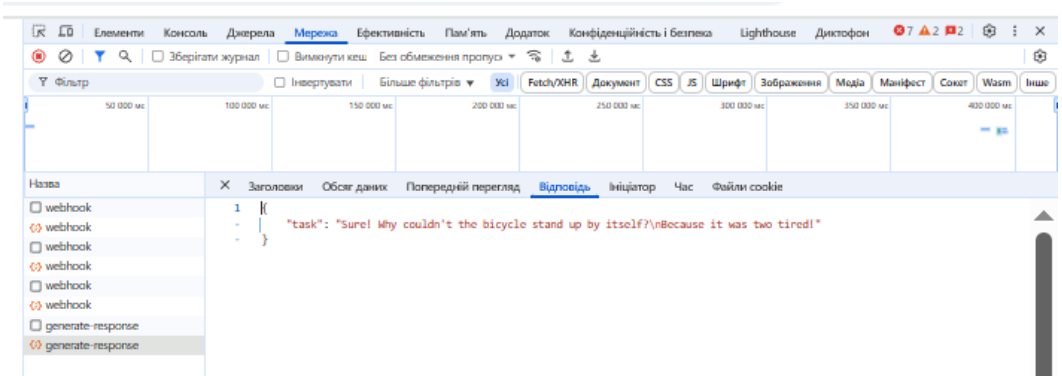


Рисунок 4.62 – Отримані дані із OpenAI

Отже, із рисунків 4.54-4.62 можна зробити висносок, що інтеграція з Rasa працює коректно.

4.3.4 Тест обробки голосу через Vosk

Перевіримо, що аудіо з фронтенду розпізнається і текст передається далі.

Кроки для тестування:

- Записати аудіо.
- Відправити на бекенд, який використовує Vosk.
- Отримати транскрипцію.
- Перевірити її коректність.

Очікується, що транскрипція буде співпадати з аудіо.

Краще всього перевірити це на сторінці гри «Tongue Twister Challenge», де потрібно вимовити скоромовку, але все ж ще можна перевірити транскрицію вцілому. Ззапишемо аудіо, наприклад, із текстом “My name is Olga and I am from Ukraine” і подивимось на запити та транскрипцію-відповідь:

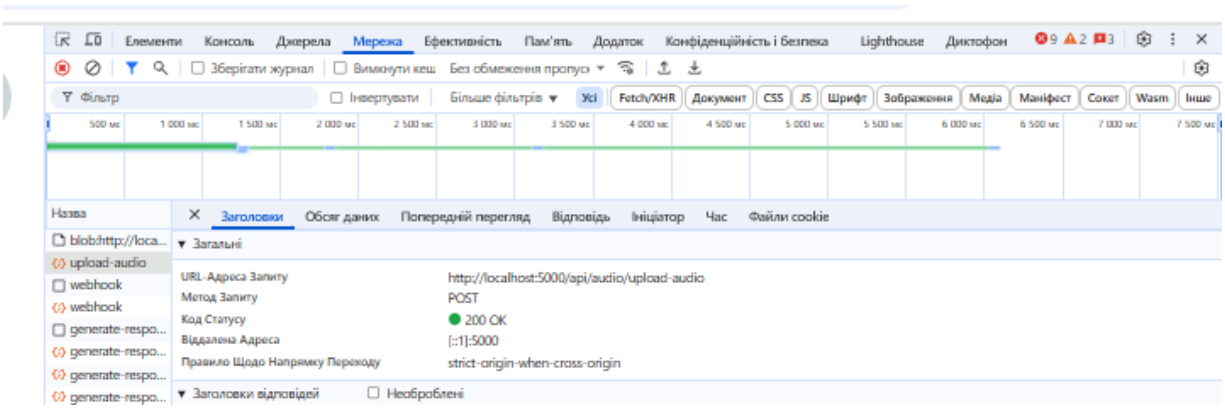


Рисунок 4.63 – Відповідь API (успіх, статус 200)

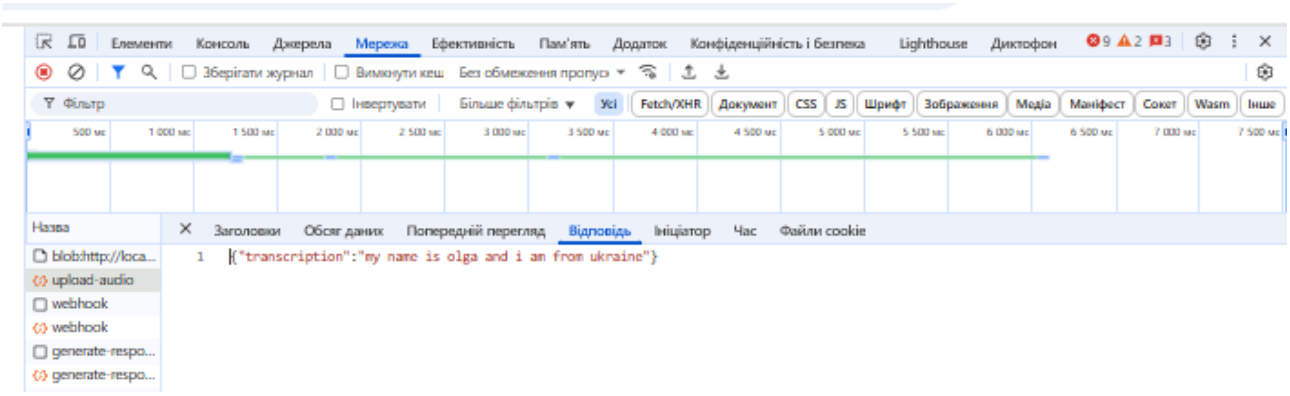


Рисунок 4.64 – Отримані дані із Vosk

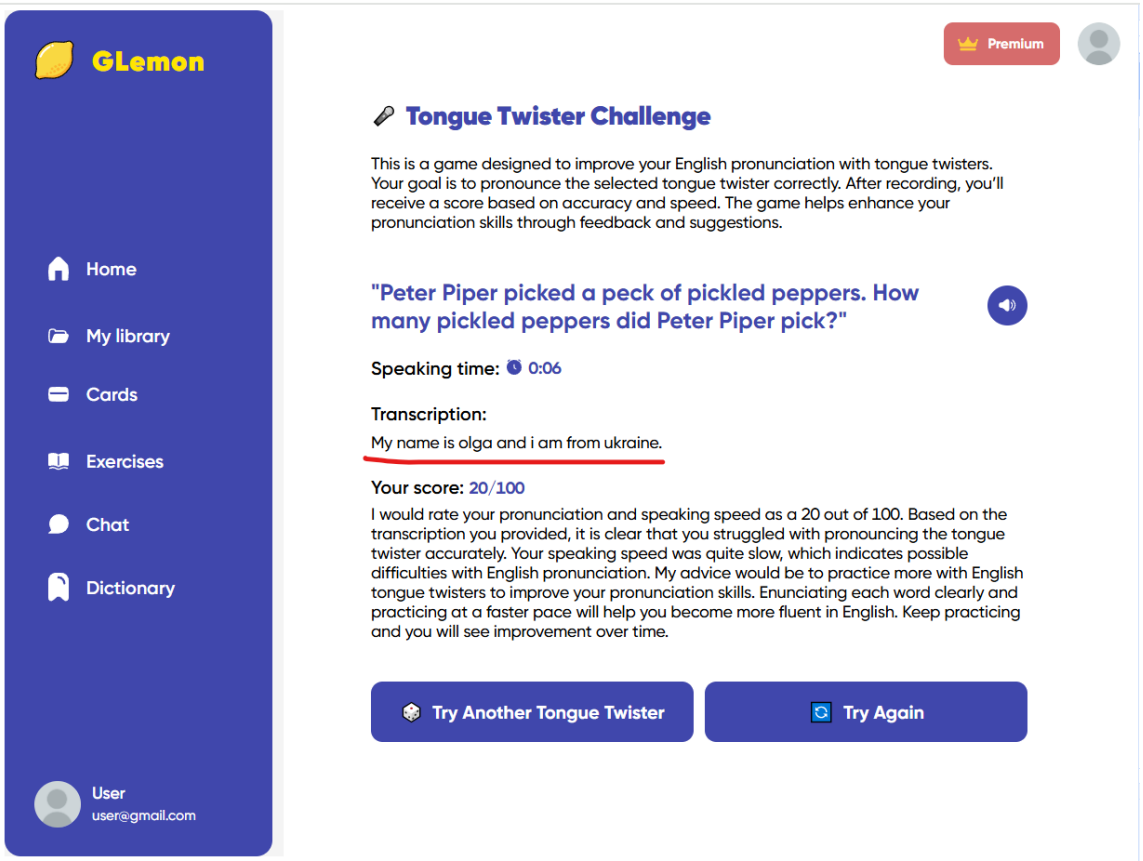


Рисунок 4.65 – Текст з аудіо після транскрибування від Vosk

Отже, із рисунків 4.63, 4.64, 4.65 видно, що голосова інтеграція працює коректно. Під час тестування обробки голосу через систему Vosk було підтверджено, що аудіо, записане з фронтенду, успішно передається на бекенд, де відбувається коректна транскрипція.

4.3.5 Результатів інтеграційних тестів

№	Тест	Очікуваний результат	Фактичний результат	Висновок
1	Завантаження, додавання та видалення слів	Слова відображаються у таблиці; слова додаються до БД і відображаються; слова видаляються і зникають з UI та БД	Успішно	Інтеграція з БД і API коректна
2	Взаємодія з OpenAI	Відповідь AI приходить і відображається	Успішно	OpenAI інтеграція коректна
3	Взаємодія з Rasa	AI відповідає або відбувається fallback на OpenAI	Успішно	Rasa і fallback працюють
4	Голосове розпізнавання (Vosk)	Транскрипція відповідає аудіо	Успішно	Голосова інтеграція працює

Таблиця 4.1 – Результати інтеграційних тестів

Отже, проведені інтеграційні тести підтвердили стабільність та коректність роботи основних компонентів системи. Взаємодія з базою даних, API, а також інтеграції з AI-сервісами OpenAI та Rasa працюють відповідно очікуванням. Голосове розпізнавання через Vosk демонструє хорошу точність транскрипції, що гарантує надійну роботу голосових функцій. Таким чином, інтеграція всіх складових системи виконана успішно та готова до подальшого використання.

4.4 Шляхи удосконалення додатка

Розроблений вебдодаток уже має добре розроблену функціональність, яка дозволяє користувачам ефективно взаємодіяти з системою, використовуючи голосові та текстові повідомлення для навчання та практики англійської мови. Проте, у майбутньому його можна покращувати, як і будь-який продукт.

Для подальшого розвитку вебдодатку можна врахувати покращення точності голосового розпізнавання, тобто використовувати більш потужні моделі для голосового розпізнавання для забезпечення точнішого транскрибування у складних або шумних умовах. А також враховувати мовні акценти та інші особливості мови користувачів для покращення точності розпізнавання.

Також у майбутньому можна додати покращення взаємодії з користувачем. Мається на увазі, наприклад, системи рекомендацій чи адаптивні відповіді, враховуючи історію чатів та вподобання користувача. Можна впровадити можливість збереження чатів для подальшого перегляду, що дозволить користувачам легко і у будь-який час повертатися до попередніх діалогів із ботом.

Також гарної ідеєю було б створити можливості співпраці із навчальними закладами, тобто створити варіанти для інтеграції з корпоративними чи освітніми платформами для збору даних та статистики по прогресу користувачів. Також можна було б розробити можливість для користувачів створювати індивідуальні навчальні курси та поширювати їх.

Усі ці заходи при впровадженні дозволять зробити користування вебдодатком ще зручнішим, ефективнішим та користішим.

ВИСНОВОК ДО РОЗДІЛУ 4

При тестуванні вебдодатку було охоплено основні аспекти його роботи, наприклад, перевірка функціональності ключових елементів інтерфейсу, таких як система реєстрації та аутентифікації юзерів, генерація та обробка запитів до зовнішніх сервісів, а також взаємодія з користувачем через голосові та текстові повідомлення. Також тестування дозволило перевірити надійність обробки даних у базі даних, а також точність відповіді створеної системи на запити користувачів.

Особливу увагу було приділено тестуванню голосових функцій додатку, оскільки інтеграція таких можливостей як розпізнавання мови, синтез мови та взаємодія з API є важливим елементом в підвищенні зручності та іноваційності вебдодатку.

Також під час тестування були виконані інтеграційні та функціональні тести, які дозволяють оцінити взаємодію між різними модулями вебдодатку, зокрема із зовнішніми сервісами, такими як Vosk, Rasa та OpenAI. Після тестування переконалися, що обробка даних коректна, зокрема у взаємодії із моделями ШІ для генерації відповідей на запити користувачів.

Отже, у результаті проведеного тестування вебдодатку було зауважено декілька рекомендацій для подальшого розвитку та масштабування розробленого продукту. Це реалізація додаткових функцій та ігор, вдосконалення інтерфейсу для зручності користувачів, покращення точності обробки голосових запитів та оптимізація швидкодії.

У підсумку, тестування довело, що розроблений вебдодаток відповідає більшості вимог й добре виконує свою основну функцію, допомагати користувачам у вивченні англійської мови через інтерактивні завдання та діалоги.

ВИСНОВКИ

У межах даного дипломного проєкту було розроблено вебдодаток для вивчення англійської мови з інтерактивними завданнями та голосовими функціями із використанням штучного інтелекту. Основними функціями системи стали автентифікація та реєстрація користувачів, інтеграція з моделями штучного інтелекту для генерації індивідуальних завдань, оцінки правильності виконання цих завдань, а також можливість взаємодії через текстові та голосові повідомлення. Користувачі можуть вибирати ігри та типи вправ на будь-який смак, наприклад, словникові завдання, вимова, граматика або чат-бот, й отримувати зворотний зв'язок, що базується на своїх відповідях.

Розробка системи почалась із аналізу існуючих рішень на ринку, що дозволило підкреслити для себе усі переваги та недоліки наявних сервісів. Усе це в сумі дало змогу сформулювати вимоги та створити функціональний та зручний сервіс для користувачів для вивчення мови. Був детально спроєктований інтерфейс, який має інтуїтивно зрозуміле та зручне управління для користувачів, а також адапотований для використання на різних пристроях.

Вебдодаток був розроблений із використанням сучасних технологій для забезпечення взаємодії з користувачем через текстові та голосові повідомлення. Тестування показало, що основні функції розробленої системи, такі як реєстрація користувачів, генерація завдань, голосові відповіді та перевірка результатів, працюють коректно.

Додатково було проведено тестування зручності користування інтерфейсом, що підтвердило гарну розробку. А функціональне тестування показало, що розроблений вебдодаток правильно генерує завдання та забезпечує зворотний зв'язок користувачам, а також коректно обробляє аудіо та текстові дані.

При аналізі та реалізації вебдодатку було сформульовано деякі можливі подальші рекомендації для розвитку та перспективи сервісу. В подальшому

можна розглядати покращення точності транскрипції голосових команд та реалізацію мультимовної підтримки, яка дозволить розширити аудиторію користувачів за рахунок інших мов, а також можливість інтеграції з іншими навчальними платформами та навіть школами.

У підсумку, можна сказати, що створений вебдодаток є потужним інструментом для вивчення англійської мови, забезпечуючи інтерактивний підхід до навчання через використання голосових та текстових завдань. Протестовані функції працюють добре, а подальший розвиток сервісу дозволить зробити додаток ще більш зручним, ефективним та корисним для все більшої аудиторії.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Mozilla Developer Network (MDN). JavaScript Guide. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide.
2. Eloquent JavaScript. Marijn Haverbeke. 3rd edition, 2018. 472 p. URL: https://eloquentjavascript.net/.
3. React Documentation. URL: https://reactjs.org/docs/getting-started.html.
4. Stefanov S. "React Up & Running". O'Reilly Media, 2016. 220 p. URL: https://dl.ebooksworld.ir/books/React.Up.and.Running.2nd.Edition.Stoyan.Stefanov.OReilly.9781492051466.EBooksWorld.ir.pdf.
5. OpenAI Documentation. URL: https://platformOpenAI.openai.com/docs/overview.
6. Rasa Documentation. URL: https://legacy-docs-oss.rasa.com/docs/rasa/.
7. Web Speech API Documentation. URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API.
8. Vosk API Documentation. URL: https://alphacephei.com/vosk/.
9. Lutz M. Learning Python. 4th ed. CA 95472: O'Reilly, 2009. 1162 p. URL: https://cfm.ehu.es/ricardo/docs/python/Learning_Python.pdf.
10. MongoDB Documentation. URL: https://www.mongodb.com/docs/.

ДОДАТОК 1

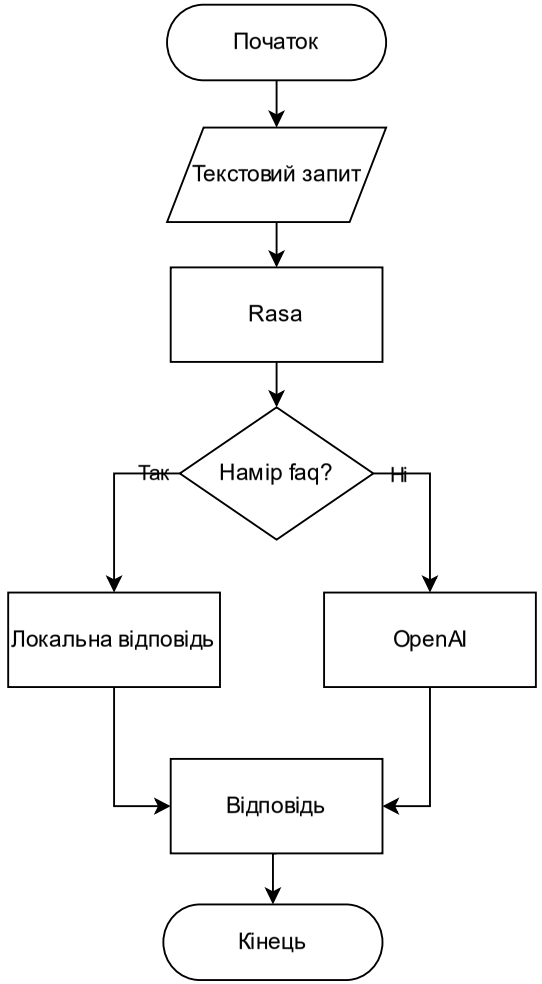
Вебзастосунок для вивчення англійської мови із використанням штучного інтелекту

Блок-схема логіки взаємодії Rasa з OpenAI

ІАЛЦ.0454400.005 Д1

Аркушів 1

Київ - 2025



Зм.	Аркуш	№докумену	Підпис	Дата	ІАЛЦ.045440.005 Д1
Розроб.		Петрук О. С.			Вебзастосунок для вивчення англійської мови з використанням штучного інтелекту
Перевір.		Петрашенко А.В.			
Т.контр.					Літера: — Маса: — Масштаб: —
Н.контр.		Клигченко Я.М.			Аркуш:1 Аркушів:1
Затв.		Романюкевич В.О.			КПІ ім.Ігоря Сікорського, ФІПМ, КВ-11

Інв.№орит. | Підпис.та дата | Взам.інв.№ | Інв.№дубл. | Підпис.та дата

ДОДАТОК 2

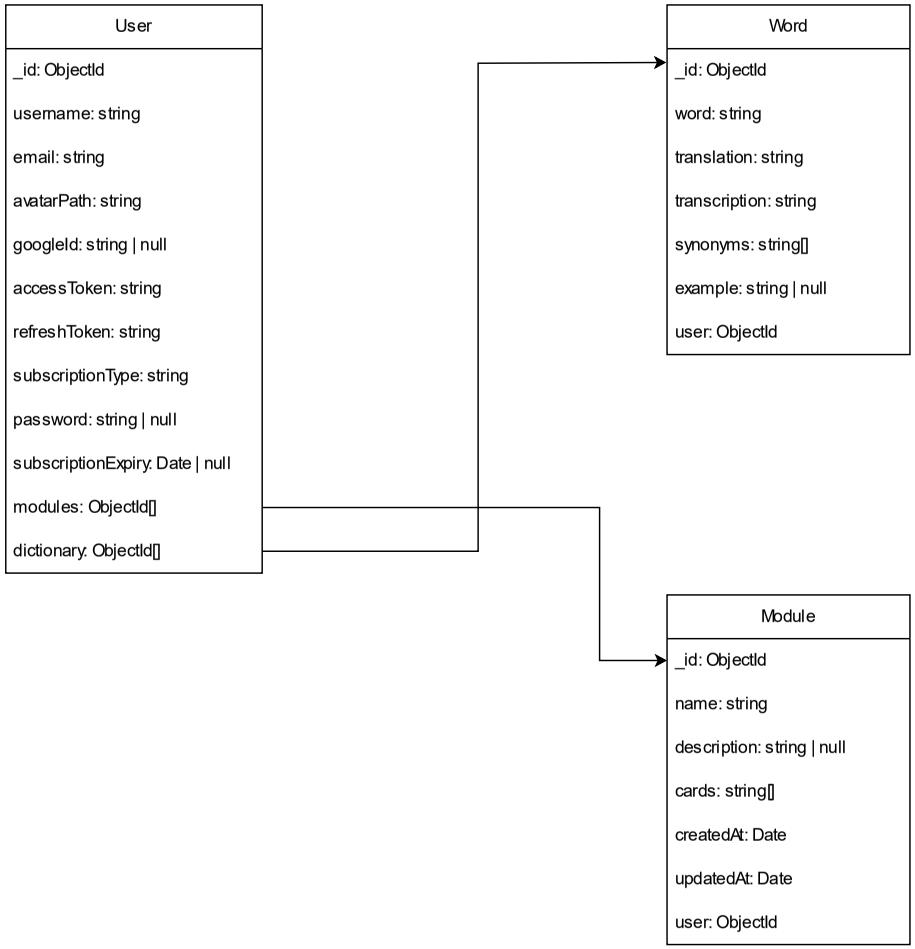
Вебзастосунок для вивчення англійської мови із використанням штучного інтелекту

Схема бази даних

ІАЛЦ.0454400.006 Д2

Аркушів 1

Київ – 2025



Text is not SVG - cannot display

						ІАЛЦ.045440.005 Д2	
Зм.	Аркуш	№документу	Підпис	Дата	Вебзастосунок для вивчення англійської мови з використанням штучного інтелекту	Літера	Маса	Масштаб
Розроб.		Пастух О.С.			*Схеми алгоритмів*			
Перевір.		Петрашенко А.В.				Аркуш 1	Аркушів 1	
Т.контр.						КПІ ім.Ігоря Сікорського, ФІПМ, КВ-11		
Н.контр.		Кличченко Я.М.						
Затв.		Романкевич В.О.						

Інв.№дубл.	Підпис і дата
Взам.інв.№	Інв.№дубл.
Підпис і дата	Взам.інв.№
Інв.№ориг.	Підпис і дата

ДОДАТОК 3

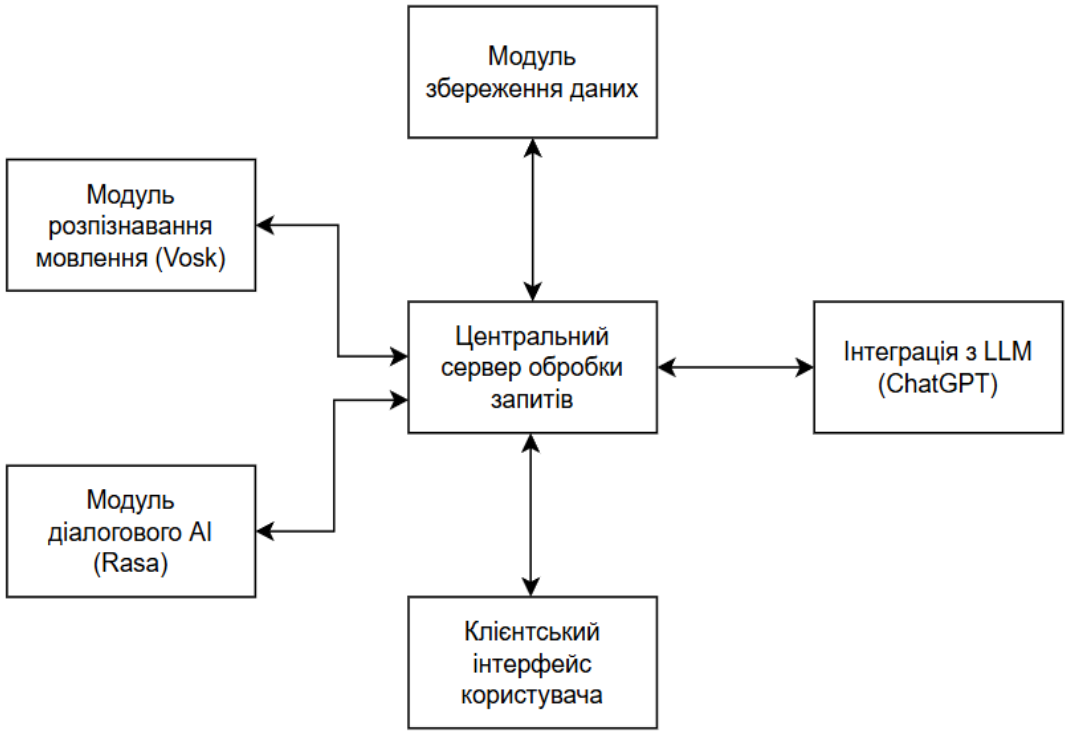
Вебзастосунок для вивчення англійської мови із використанням штучного інтелекту

Архітектура взаємодії вебдодатку

ІАЛЦ.0454400.007 Д3

Аркушів 1

Київ – 2025



Зм.	Аркуш	№ документу	Підпис	Дата	ІАЛЦ.045440.005 Д3
Розроб.		Петрук О.С.			Вебзастосунок для вивчення англійської мови з використанням штучного інтелекту *Архітектура взаємодії вебдодатку*
Перевір.		Петрашенко А.В.			
Т.контр.					
Н.контр.		Клигченко Я.М.			
Затв.		Романкевич В.О.			

Літера	Маса	Масштаб
Аркуш 1	Аркушів 1	

КПІ ім. Ігоря Сікорського, ФІПМ, КВ-11

Інв.№ оригін. | Підпис і дата | Взам.інв.№ | Інв.№ дубл. | Підпис і дата

ДОДАТОК 4

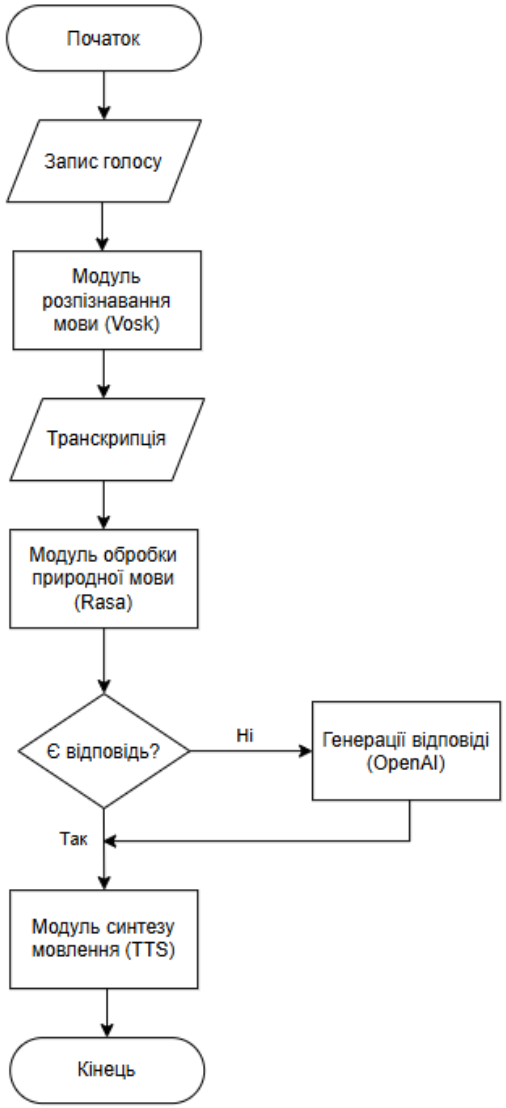
Вебзастосунок для вивчення англійської мови із використанням штучного інтелекту

Блок-схема алгоритму роботи голосового AI-асистента

ІАЛЦ.0454400.008 Д4

Аркушів 1

Київ – 2025



Зм.	Аркуш	№ документу	Підпис	Дата	ІАЛЦ.045440.005 Д4		
Розроб.		Петрук О.С.			Вебастосунок для вивчення англійської мови з використанням штучного інтелекту	Літера	Маса	Масштаб
Перевір.		Петрищенко А.Ю.						
Т.контр.					Схема роботи голосового AI-асистента	Аркуш:1	Аркушів:1	
Н.контр.		Клинченко Я.М.				КПІ ім. Ігоря Сікорського, ФІПМ, КВ-11		
Затв.		Романкевич В.О.						

Інв.№ориг.	Підписи та дата	Взам.інв.№	Інв.№дубл.	Підпис та дата