МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

ДІДУХ Данило Романович

Програмний модуль для вивчення іноземної мови на основі штучного інтелекту / Software Module for Learning Foreign Languages Based on Artificial Intelligence

спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-41
Д.Р. Дідух
_____

Науковий керівник
В.І. Дорош
_____

Кваліфікаційну роботу допущено до захисту
«___»_____ 20___ р.

Завідувач кафедри
_____ М.П. Комар

ТЕРНОПІЛЬ – 2024

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М.П. Комар
«____» _____2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Дідух Данило Романович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Програмний модуль для вивчення іноземної мови на основі штучного інтелекту / Software Module for Learning Foreign Languages Based on Artificial Intelligence
керівник роботи Дорош Віталій Іванович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 12 грудня 2023 р. № 753.
2. Строк подання студентом закінченої кваліфікаційної роботи 15 травня 2024 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити:
- провести опис предметної області;
- зробити аналіз використання штучного інтелекту у вивчені іноземних мов;
- дослідити стан розвитку методів вивчення іноземних мов;
- зробити постановку задачі дослідження;
- дослідити поточний стан та аналіз сфери вивчення іноземних мов на основі штучного інтелекту;
- дослідити теоретичні основи застосування штучного інтелекту у вивченні іноземних мов;
- розробити модель генерації речень на основі бажаних слів з використанням мікро-сервісної архітектури;
- налаштувати процес генерації речень використовуючи штучний інтелект;
- зробити аналіз відповідності та коректності згенерованих речень;

5. Перелік графічного матеріалу в роботі:
- схема алгоритму роботи модуля;
- UML-діаграма класів.

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання 12 грудня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи.	до 01.01. 2024 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.03. 2024 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.04.2024 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2024 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2024 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів.	до 20.05.2024 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту у системі «Unicheck».	до 10.06.2024 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 14.06.2024 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії.	до 14.06. 2024 р.	

Студент_____ Д.Р. Дідух
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ В.І. Дорош
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Програмний модуль для вивчення іноземної мови на основі штучного інтелекту» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом в 55 сторінок і містить 19 ілюстрацій, 4 додатки та 25 використаних джерел.

Метою роботи є дослідження методів вивчення іноземних мов та розроблення програмного модуля для вивчення іноземної мови на основі штучного інтелекту на мові програмування Swift.

Методом розроблення обрано каскадну модель для передбачення послідовного виконання кожного етапу проекту: аналізу вимог, проектування, реалізація, тестування, розгортання та обслуговування.

Внаслідок виконання роботи обґрунтовано раціональний підхід до розроблення програмного модуля для вивчення іноземної мови на основі штучного інтелекту на основі мови програмування Swift, який дає змогу вивчати та досліджувати лексику іноземних мов.

Програмний модуль може активно використовуватись звичайними користувачами, студентами та учнями які прагнуть вивчати лексику інозених мов.

Ключові слова: ПРОГРАМНИЙ МОДУЛЬ, ІНОЗЕМНІ МОВИ, ШТУЧНИЙ ІНТЕЛЕКТ, ВИВЧЕННЯ, IOS-РОЗРОБКА.

ANNOTATION

Qualification work on the topic "Software module for learning a foreign language using artificial intelligence" for Bachelor's degree on speciality 122 «Computer Science» educational and professional program «Computer Science» is written on 55 pages and it contains 19 figures 4 annexes and 25 sources.

The purpose of the work is to study methods of learning foreign languages and develop a software module for learning a foreign language using artificial intelligence in the Swift programming language.

The cascade model was chosen as the development method to provide for the sequential execution of each stage of the project: requirements analysis, design, implementation, testing, deployment and maintenance.

As a result of the work, a rational approach to the development of a software module for learning a foreign language using artificial intelligence based on the Swift programming language, which allows learning and researching foreign language vocabulary, is substantiated.

The software module can be actively used by ordinary users, students and pupils who want to learn foreign language vocabulary.

Keywords: SOFTWARE MODULE, FOREIGN LANGUAGE, ARTIFICIAL INTELLIGENCE, LEARNING, IOS-DEVELOPMENT

ЗМІСТ

ВСТУП

Актуальність теми: обґрунтовується кількома важливими аспектами. По-перше, зростаюча глобалізація і міжнародна співпраця вимагають вільного володіння декількома мовами, а традиційні методи навчання часто не задовольняють потреби сучасних користувачів. По-друге, швидкий розвиток технологій мобільного та електронного навчання створює попит на інтерактивні, доступні та ефективні програми. Інтеграція різних технологій штучного інтелекту, таких як природна мовна обробка і машинне навчання, дозволяє створювати більш реалістичні імітаційні середовища для практики мовних навичок, що значно покращує розуміння і засвоєння матеріалу. В умовах дистанційного навчання та обмеженого доступу до традиційних мовних курсів, такі програмні рішення стають незамінними інструментами для самостійного навчання. Загалом, розробка програмного модуля для вивчення іноземної мови на основі штучного інтелекту відповідає сучасним викликам і потребам в галузі освіти, сприяючи не тільки підвищенню якості освіти, але й збільшенню доступу до неї для широкого кола людей.

У цій роботі створення інтерактивного інтерфейсу стає важливим аспектом розробки. Він допоможе здійснювати введення даних та отримання зворотного зв’язку від додатку. Його основна мета це максимально простий інтерфейс який забезпечить максимально швидку взаємодію користувача з програмним модулем.

Мета полягає у створені програмного модуля на мові Swift, який дозволить користувачам ефективно вивчати іноземну лексику через контекстуалізоване використання слів у реченнях. Модуль буде використовувати можливості мікро-сервісів, таких як GPT 3.5 та DeepL, для генерації прикладів речень з різними контекстами, що дозволить користувачам глибше зрозуміти вживання слова у речені.

Для досягнення мети необхідно вирішити такі задачі:

- Провести аналіз предметної області вивчення іноземних мов.
- Дослідити найефективніші методи вивчення іноземних мов.
- Розробити архітектуру програмного модуля.

- Реалізувати програмний модуль.
- Провести тестування та оптимізацію модуля.

Зазначені завдання дозволять успішно реалізувати поставлену мету і створити ефективний та сучасний програмний модуль для вивчення іноземної мови на основі штучного інтелекту.

Об’єкт дослідження є програмний модуль для вивчення іноземних мов.

Предмет дослідження розробки є програмний модуль для вивчення іноземної мови на основі штучного інтелекту та інтерактийвний користувацький інтерфейс.

Апробація результатів дослідження. Основні теоретичні положення роботи й практичні результати дослідження доповідалися й обговорювалися: «ПРОГРАМНИЙ МОДУЛЬ ДЛЯ ВИВЧЕННЯ ІНОЗЕМНОЇ МОВИ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ» VI Всеукраїнської студентської наукової конференції «Експериментальні та теоретичні дослідження в контексті сучасної науки» (21.06.2024, м. Рівне, Україна).

1 АНАЛІЗ ВИВЧЕННЯ ІНОЗЕМНОЇ МОВИ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ

1.1 Опис предметної області вивчення іноземної мови на основі штучного інтелекту

До основного складу функцій бізнес процесу потрібно віднести функцію створення контенту. Програмний модуль використовує генерацію речень в яких використовуються задані користувачем слова іноземної лексики. В самому модулі використовується мікро-сервіс GPT 3.5, для створення різномнанітних прикладів речень. Модель GPT тренується на величезному дата-сеті інтернет текстів, які написані на різних мовах. Паралельно модель навчається передбачати наступне слово у реченні, базуючись на контексті попередніх слів. Саме це дає змогу їй зрозуміти структуру, граматику та значення слів. Модель обробляє завдання яке написане природною мовою, його ще називають «Prompt», який вводиться для отримання потрібної відповіді від мікро-сервісу. В наслідок натренованості GPT на великих об’ємах текстів, він може генерувати свої відповіді різними мовами, адаптуючись до різних мовних особливостей, зазначається у праці [22].

Також однією з основних функцій програмного модуля є переклад згенерованих штучним інтелектом контекстуалізованих речень. Для перекладу використовується мікро-сервіс DeepL це один з найпотужніших та найточніших інструментів для перекладу, який використовує технології глибокого навчання та штучного інтелекту для перекладу текстів. Для тренування своєї моделі він використовує великі обсяги двомовних текстів. До цих текстів відносяться веб-сторінки, книги та статті. Як зазначається у праці [25], процес перекладу відбувається коли на мікро-сервіс DeepL приходить запит з згенерованим текстом, який розбирається на токени і подається на вхід моделі. Модель, використовуючи свої знання, які сформувались в процесі навчання, обробляє текст та передає перекладений текст на вихід.

До основних функцій також входять тестування та оптимізації програмного модуля. Для тестування використовується програмне середовище Xcode у якому присутній інструмент симуляції додатку в операційній системі IOS. Створюються

тестові набори слів для перевірки згенерованих речень та їх перекладу на відповідність контексту та правильному використанню слів. Під час процесу оптимізації виконується перевірка на витоки пам’яті та оптимізується алгоритм генерації та перекладу речень.

Функція аналізу результатів включає в себе збір статистики та зворотного зв’язку щодо досвіду використання модуля користувачами. Також користувачами буде оцінюватись ефективність в вивчені іноземної мови.

У діаграмі дерева функцій (рисунок 1.1) відображаються основні функції та підфункції, це допомагає полегшити розуміння структури за взаємозв’язків між окремими частинами модуля.

Рисунок 1.1 – Діаграма дерева функцій

Для детальнішого розуміння потоку даних у програмному модулі використовується діаграма моделі предметної області (рисунок 1.2)

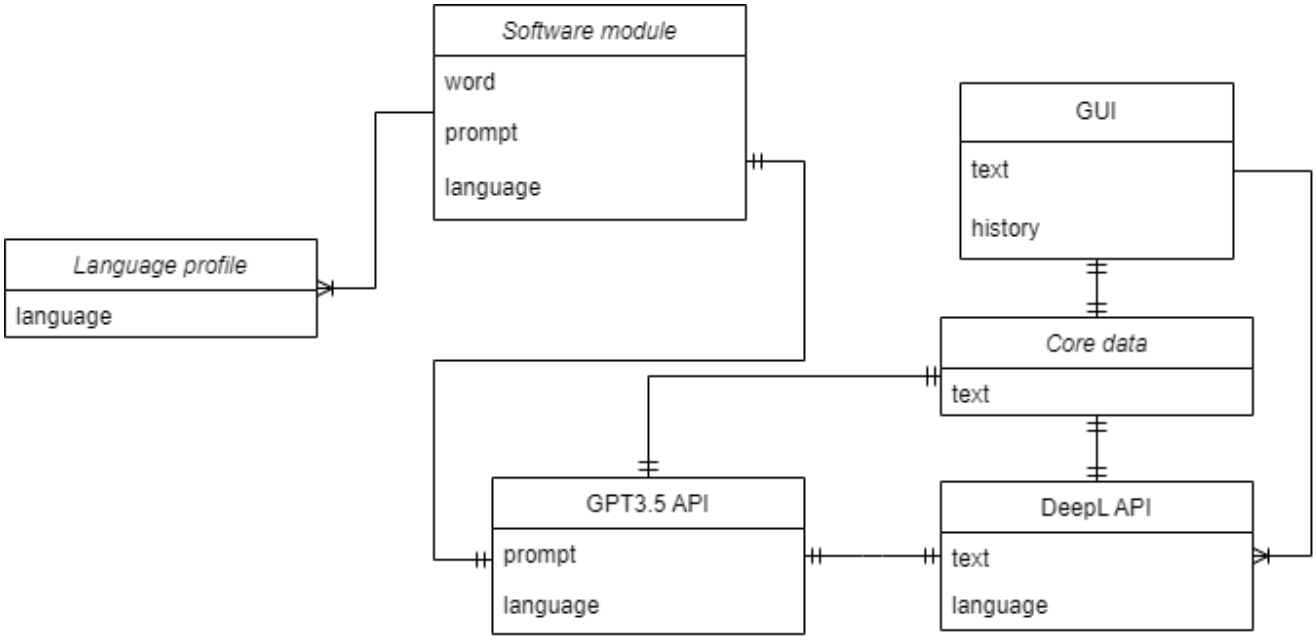


Рисунок 1.2 – Модель предметної області

1.2 Огляд і аналіз існуючих аналогів

Варто також звернути увагу на вже існуючі аналоги які реалізують функції вивчення іноземної мови на основі штучного інтелекту.

DeepL – це потужний онлайн-перекладач, який використовує новітні алгоритми штучного інтелекту для забезпечення точних та високоякісних перекладів.

До функцій DeepL варто включити підтримку багатьох мов для перекладу, спочатку він міг переклади сімома європейськими мовами, та згодом його можливості розширили до 31 мови. До переваг також відноситься можливість перегляду альтернативних перекладів. Функція альтернативного перекладу дозволяє користувачу вибирати інші варіанти перекладу для окремих слів або висловів. Що дає змогу натиснути на слово або фразу під час перегляду перекладу та вибрати той варіант, який найкраще відповідає реченню, оскільки DeepL відображає кілька альтернативних варіантів перекладу для даного контексту. Якщо потрібно змінити переклад, є можливість редагувати текст безпосередньо у полі введення. Також, після редагування, онлайн-перекладач покаже альтернативні варіанти для відредагованого тексту. Під час редагування перекладу з’являється можливість вибрати альтернативний варіант для відредагованого тексту. Потрібно

лише натиснути на слово або фразу яка редагується і обрати бажаний варіант. Функція API для інтеграції з іншими програмами та сервісами, саме ця функція дозволяє інтегрувати одну з найкращих технологій машинного перекладу в власні продукти та платформи. До можливостей API DeepL входить переклад файлів різних форматів, захист даних, а саме видалення всіх текстів після завершення процесу перекладу, що підвищує конфіденційність. Також до вмінь прикладного програмного інтерфейсу можна віднести вибір формального або неформального тону яким буде відбуватись переклад та автоматичне розпізнавання мов.

Проводячи аналіз інтерфейсу DeepL стало зрозуміло, що основною метою створеного дизайну була простота та інтуїтивно зрозумілий інтерфейс. В головному екрані користувач вводить або копіює текст для перекладу. Поруч з ним відображаються мовні налаштування та кнопка для здійснення перекладу. Перекладений текст з'являється на екрані поруч з оригіналом. Користувач може редагувати перекладений текст. Під час перекладу користувач може виділити окреме слово і отримати детальну інформацію про його варіанти перекладу.

Reverso Context – це онлайн-інструмент, який поєднує у собі функції перекладу та вивчення мови на основі контекстуалізованих прикладів.

До функцій відноситься переклад фраз і речень з урахуванням контексту. Цей інструмент має базу даних з перекладами слів та виразів у різних контекстах, вона сформована на основі реальних документів, субтитрів фільмів та різних статей. Присутня функція вимови, вона створена для відтворення та прослуховування правильної вимови слів. Для відтворення правильної вимови використовуються інструменти штучного інтелекту, вони створенні на основі фонетичних правил з допомогою яких створюється розуміння мовних закономірностей, також, моделі навчаються на великій кількості аудіозаписів намагаючись відтворити вимову носіїв мови використовуючи транскрипцію. Також є функція для вивчення нових слів та виразів шляхом додавання їх у особистий словник. Це дозволяє системі оновлювати та добавляти нові слова.

Інтерфейс Reverso Context зручний та орієнтований на швидкий доступ до функцій. На головній сторінці знаходиться поле для введення тексту, що потребує перекладу чи пояснення. Поруч знаходяться мовні налаштування і кнопка

перекладу. На екрані результатів знаходиться перекладений текст з додатковими контекстними прикладами використання. Користувач може переглядати та слухати ці приклади, а також зберігати їх у словниковий запас. Функція словнику для пошуку і збереження нових слів і виразів, доступ до особистого словника.

Langua є новаторським додатком, який використовує штучний інтелект для створення персоналізованих планів навчання та генерації контекстуалізованих вправ.

До функцій відноситься персоналізовані рекомендації щодо вивчення слів. Ці рекомендації базуються на аналізі раніше виконаних вправ у яких виникли труднощі. Функція яка створює вправи на основі контекстуалізованих прикладів. Вона використовує штучний інтелект для генерації та підбору вправ. Функція тестування рівня знання мови та адаптація уроків. Ця функція виконується за допомогою різних вправ які будуть використанні для перевірки знань та визначення рівня знання іноземної мови. Функція інтеграції з такими інструментами перекладу як DeepL. Додаток використовує API інтеграцію для покращення якості перекладу, розуміння контексту та генерації вправ.

Інтерфейс Langua функціональний і сучасний, орієнтований на швидкість та зручність для користувача. На головному екрані відображається прогрес навчання і рекомендації на основі попередньої активності. Вкладка вправ включає інтерактивні завдання з контекстуалізованими прикладами. Розділ статистики відображає детальну статистику навчання, сильні та слабкі сторони користувача.

Проаналізовані програми DeepL, Reverso Context та Langua мають свої унікальні переваги та функції, що робить їх ефективними інструментами для вивчення іноземної мови.

DeepL – основний акцент робиться на високоякісному перекладі текстів. Програма забезпечує швидкий та точний переклад, що ідеально підходить для людей які використовують його як професійний інструмент, студентів та всіх, хто потребує точних перекладів на різні мови.

Reverso Context – ефективний інструмент для вивчення мови через контекстуалізовані приклади. Підходить для тих, хто хоче глибше розуміти, як використовуються слова і фрази в різних ситуаціях. Інтерактивність і можливість

збереження значень у словниковий запас значно допомагають у запам'ятовуванні та практичному застосуванні нової лексики.

Langua – зосереджує свої зусилля на персоналізованому навчанні та використанні контекстуалізованого словника. Це особливо корисно для тих, хто хоче застосовувати нові слова та фрази в реальних життєвих ситуаціях, покращуючи тим самим свою комунікацію через вправи на основі штучного інтелекту.

Кожен програмний продукт має свої сильні сторони і підходить для різних категорій учнів. Вони показують, як можна залучити сучасні технології, зокрема штучний інтелект, для покращення ефективності вивчення іноземниї мови.

Для програмного модуля можна взяти за основу кращі практики з кожного з цих продуктів: високоякісний переклад від DeepL, контекстуалізовані приклади від Reverso Context та інтеграція з інструментами від Langua. Це дозволить створити збалансований та ефективний інструмент для вивчення іноземних мов.

Можна використати мікро-сервіси GPT 3.5 та DeepL для створення прикладів речень у різних контекстах. Використати можливість збереження нових слів і фраз для подальшого вивчення та повторення.

Ці компоненти допоможуть створити конкурентоспроможний продукт, який об'єднає кращі риси існуючих рішень і запропонує користувачам унікальний та ефективний спосіб вивчати іноземну лексику в контексті.

1.3 Постановка задачі дослідження

На основі аналізу існуючих аналогів, таких як DeepL, Reverso Context та Langua, формулюється загальна концепція, а також вимоги до проєктованої системи. Впровадження необхідних структурних, функціональних та конструктивних змін дозволить підвищити ефективність вирішення завдань вивчення іноземної лекски через контекстуалізоване використання слів у реченнях.

До загальної концепції входить побудова програмного модуля, який використовує штучний інтелект для генерації та перекладу контекстуалізованих речень, з метою покращення та полегшення процесу вивчення іноземної мови.

Модуль має забезпечити користувачів інтерактивними інструментами для розуміння використання нової лексики в різних ситуаціях.

Формування вимог до проєктованої системи є ключовим етапом, який визначає її майбутню ефективність та зручність використання. Це дозволить побудувати конкурентоспроможний продукт, що відповідатиме потребам користувачів і враховуватиме найкращі практики та технології, що використовуються в сучасних аналогах.

Інтеграція мікро-сервісів у проєктований модуль є ключовим аспектом, що дозволяє забезпечити потужність і гнучкість системи. Використання GPT 3.5 для генерації контекстуалізованих речень забезпечує створення високоякісних та реалістичних прикладів використання слів у різних контекстах. Цей мікро-сервіс використовує передові алгоритми машинного навчання, які здатні розуміти контекст і створювати точні та релевантні речення, що допомагає користувачам краще зрозуміти вживання нової лексики.

Одночасно, мікро-сервіс DeepL відповідає за переклад згенерованих GPT речень на необхідну мову, забезпечуючи високоякісні та точні переклади. Інтеграція DeepL дозволяє отримувати не тільки перекладені речення, але й переглядати альтернативні варіанти перекладу, що поглиблює розуміння та навчання користувача. Завдяки такій комбінації мікро-сервісів, модуль здатний автоматично генерувати і перекладати контекстуалізовані речення у режимі реального часу, що робить процес вивчення мови більш ефективним і зручним. Ця інтеграція також дозволяє масштабувати функціональність модуля та легко додавати підтримку нових мов або вдосконалювати вже існуючі можливості, забезпечуючи постійний розвиток та модернізацію системи.

Використання мікро-сервісної архітектури є найкращим рішенням для подібного програмного модуля. В джерелах [16, 17, 20] зазначено, що архітектура мікро-сервісів – це підхід до розробки додатків, де сервіси мають невеликий розмір і розгортаються незалежно. Кожен сервіс в рамках фреймворку має незалежність, що сприяє легкості обслуговування та розробки. До переваг використання цієї архітектури можна віднести масштабованість, а саме гнучкість в масштабуванні окремих сервісів, це дозволить обслуговувати більше клієнтів. Також можливість

паралельної розробки, коли розробник може працювати над окремими сервісами одночасно. Перевагою можна вважати незалежність, коли кожен мікро-сервіс може оновлюватись окремо без злиття з усією системою, що спрощує обслуговування та розробку модуля. Разом із тим, наявність можливосі декомпозиції дозволяє розділити програмний модуль на менші частини, це полегшує розуміння та управління системою.

До функціональних вимог входить генерація контекстуалізованих речень та вона є однією з ключових функцій модуля, яка забезпечує автоматичне створення прикладів речень з використанням обраних слів у різних контекстах. Завдяки інтеграції з мікро-сервісом GPT 3.5, модуль здатний обробляти введені користувачем слова і генерувати релевантні речення, що відображають реальні ситуації та різноманітні аспекти застосування цих слів. Це дозволяє користувачам побачити слово в дії, зрозуміти його значення та правильне використання в різних мовних контекстах. Додатково, модуль надає кілька варіантів використання слів, беручи до уваги різні обставини та стилі мовлення, такі як формальний, неформальний, технічний чи розмовний. Це допомагає користувачам розширити своє розуміння мовних форм і функцій, дізнатися про синоніми, фразеологічні вирази та інші мовні явища, які можуть бути корисні в різних ситуаціях. Таким чином, генерація контекстуалізованих речень не лише сприяє ефективному вивченню нової лексики, але й допомагає користувачам стати більш гнучкими і впевненими в своїх мовних навичках, здатними легко перемикатися між стилями мовлення і адаптуватися до різних комунікативних ситуацій.

Переклад речень є важливою функцією модуля, що дозволяє користувачам отримувати переклади на обрану мову для глибшого розуміння та засвоєння матеріалу. Використання передових технологій, як-от DeepL, забезпечує високу точність і якість перекладів, тим самим допомагаючи користувачам знайомитися з новою лексикою та граматичними конструкціями в різнотипних контекстах. Додатково, модуль надає можливість швидкого перемикання між оригінальним текстом і його перекладом, що значно полегшує процес навчання. Ця функція дозволяє користувачам легко порівнювати оригінальний і перекладений текст, оперативно підвищуючи їхню здатність до розуміння і запам'ятовування нових слів

та фраз. Швидке перемикання також сприяє розвитку навичок читання і аудіювання, допомагаючи користувачам відстежувати структуру речень та інші мовні елементи у різних мовах.

Вимога надання інтерактивної взаємодії користувачу, а саме, надання користувачам можливості редагування, збереження та повторного використання речень. Запровадження функції прослуховування вимови речень. Ця функція покращує навички сприйняття на слух та розуміння правильної вимови користувача. Система використовує технологію перетворення тексту у мову для створення аудіофайлу, застосовуючи високоякісні TTS алгоритми, такі як Google Text-to-Speech або Amazon Polly, для забезпечення природного та чіткого вимовлення. Користувач натискає кнопку відтворення і слухає речення, при цьому маючи можливість повторювати відтворення стільки разів, скільки потрібно для повного розуміння та відпрацювання вимови.

До конструктивних вимог відноситься зручний та інтуїтивний інтерфейс. Зручний та інтуїтивний інтерфейс забезпечує користувачам легкість у виконанні завдань та отриманні зворотного зв'язку завдяки добре продуманому дизайну, орієнтованому на користувача. Інтерфейс спроектовано таким чином, щоб користувачі могли швидко і без зусиль знаходити та використовувати всі доступні функції. У праці [24] зазначенно про використання зручних вкладок та панелей інструментів які забезпечують швидкий доступ до основних функціональних можливостей модуля, що дозволяє ефективно навчатись, не відволікаючись на складні навігаційні елементи. Це включає логічно структуровані розділи для вводу тексту, перегляду результатів перекладу, збереження та управління історією записів.

Оптимізація продуктивності модуля передбачає зменшення часу відгуку за рахунок оптимізації алгоритмів та ефективного використання ресурсів. Це досягається завдяки застосуванню алгоритмів з високою обчислювальною ефективністю, які мінімізують затримки при обробці запитів користувача та генерації перекладів або прикладів речень. Використання лише необхідних ресурсів та їхній розумний розподіл сприяє швидкому і надійному виконанню завдань без зайвих витрат енергії або процесорного часу. Крім того, забезпечення стійкої роботи навіть при значному навантаженні включає масштабування

серверних ресурсів, щоб гарантувати безперервну доступність та стабільну продуктивність модуля під час збільшеної кількості користувачів або обробки складних завдань. Інженерні рішення щодо балансування навантаження та оптимізації кешування також є ключовими для підтримки стабільної роботи і швидкого відгуку системи.

Мультиплатформеність модуля передбачає підтримку різних операційних систем, таких як iOS та macOS, для забезпечення доступності широкій аудиторії користувачів. Це означає, що модуль повинен бути розроблений з урахуванням особливостей кожної платформи, що включає адаптацію інтерфейсу та функціональності для забезпечення зручності і стабільної роботи на різних пристроях, від смартфонів до настільних комп'ютерів. Використання кросплатформних фреймворків, таких як SwiftUI або інших відповідних інструментів, дозволяє створити уніфікований код, який полегшує підтримку і оновлення модуля, а також забезпечує узгоджений користувацький досвід незалежно від платформи. Це також сприяє зниженню витрат на розробку і підтримку, дозволяючи розробникам одночасно впроваджувати нові функції та виправлення для всіх підтримуваних пристроїв. Мультиплатформена стратегія гарантує, що всі користувачі зможуть користуватися перевагами модуля, незалежно від того, яким пристроєм вони користуються, забезпечуючи таким чином максимальне охоплення та залученість.

Впровадження зазначених вимог дозволить створити інноваційний програмний модуль для вивчення іноземної лекски через контекстуалізоване використання слів у реченнях. Це сприятиме не тільки покращенню якості навчання користувачів, але й забезпечить високу продуктивність та надійність роботи модуля.

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ МОДУЛЯ

2.1 Загальна структура проєктованої системи (модуля)

Метою цього підрозділу є викладення концептуальних засад вирішення поставленої задачі. В цьому розділі представлена функціональна структура розроблюваного модуля, перелік його функцій та їхні характеристики. Описуються інформаційні зв’язки між елементами системи, а також визначаються зв’язки з іншими модулями та зовнішніми об’єктами, які є джерелами або споживачами інформації.

Функціональна структура системи складається основних компонентів, які виконують роль генерації, обробки та отримання вхідних даних для забезпеченні роботи модуля.

Користувацький інтерфейс є одним з ключових компонентів проєктованої системи, що забезпечує взаємодію користувача з модулем. Інтерфейс дозволяє здійснювати ввід та вивід тексту, що є базовою функцією для початку роботи з системою. Користувачі можуть легко вводити слова або фрази, які вони хочуть перекласти або отримати приклади вживання, а також переглядати результати перекладу та згенеровані контекстуалізовані речення. Інтерфейс також забезпечує управління налаштуваннями перекладу і генерації речень, надаючи користувачам можливість вибирати мову перекладу. Це робить процес налаштування зручним для кожного користувача.

Окрім цього, інтерфейс відповідає за візуалізацію результатів перекладу та контекстуалізованих прикладів, роблячи їх зрозумілими і доступними. На екрані користувачі можуть бачити поруч оригінальний текст та його переклад, з можливістю швидкого перемикання між ними для полегшення навчання. Контекстуалізовані приклади виводяться так, щоб користувач міг легко зрозуміти різні варіанти використання слова або фрази в різних контекстах. Інтерфейс також може включати додаткові візуальні елементи, такі як підсвічування ключових слів, інтерактивні підказки та інші елементи, що сприяють кращому засвоєнню матеріалу. Це робить користувацький досвід максимально інтуїтивним і

ефективним, забезпечуючи високу залученість і задоволеність від використання модуля.

Модуль генерації речень є важливим компонентом проєктованої системи, що забезпечує автоматичне створення контекстуалізованих речень за допомогою GPT 3.5. Цей модуль використовує передові алгоритми машинного навчання і обробки природної мови, щоб створювати речення, які ілюструють правильне використання обраних користувачем слів у різних контекстах. Завдяки можливостям GPT 3.5, модуль здатний генерувати речення, які відповідають реальним мовним ситуаціям, допомагаючи користувачам краще зрозуміти як використовувати нову лексику на практиці. Крім того, модуль надає декілька варіантів використання слів в залежності від обставин та стилю. Це включає формальний і неформальний стилі, а також технічний, розмовний та інші варіанти, що можуть бути релевантними в різних ситуаціях. Такий підхід дозволяє користувачам побачити, як слова змінюються залежно від контексту, і як правильно вживати їх у різних мовних середовищах. Користувачі можуть отримувати різнопланові приклади для одного і того ж слова, що сприяє глибшому розумінню і кращому запам’ятовуванню. Ця персоналізована та контекстуалізована інформація робить навчання ефективнішим і дозволяє користувачам швидше адаптуватися до нових мовних умов, забезпечуючи більш гнучке та впевнене використання нової лексики.

Модуль перекладу є ключовою частиною функціональної структури системи, яка забезпечує переклад згенерованих речень на необхідну мову за допомогою DeepL. Цей мікро-сервіс використовує передові алгоритми глибинного навчання для забезпечення високоякісного та точного перекладу текстів. Завдяки інтеграції з DeepL, модуль здатний автоматично перекладати згенеровані GPT 3.5 контекстуалізовані речення на обрану користувачем мову, як зазначено в праці [9], що це значно підвищує зручність і ефективність навчання.

Модуль перекладу також надає можливість користувачам швидко перемикатися між оригінальним і перекладеним текстом. Це функціональне рішення дозволяє користувачам легко порівнювати вихідний текст з його перекладеною версією, що сприяє кращому розумінню і засвоєнню нової лексики. Така можливість особливо корисна під час навчання, оскільки дозволяє оперативно

перевіряти правильність перекладу, розуміти значення слів і фраз в обох мовах, а також вивчати різні варіанти перекладу одного і того ж речення. Перемикання між текстами здійснюється через зручний і інтуїтивний інтерфейс, що робить процес навчання більш інтерактивним і адаптивним до потреб користувача. Завдяки цьому функціоналу, модуль перекладу сприяє глибокому та ефективному вивченню іноземної мови, забезпечуючи користувачів інструментами для повноцінного занурення у двомовний контекст.

Модуль збереження даних є важливим компонентом функціональної структури системи, відповідальним за зберігання та управління усіма даними, що вводяться або генеруються під час роботи користувача з модулем. Він зберігає введені користувачем слова, згенеровані GPT 3.5 контекстуалізовані речення та перекладені DeepL речення, забезпечуючи надійну і безпечну базу даних для доступу у майбутньому. Завдяки цьому користувачі можуть з легкістю повертатися до раніше опрацьованих матеріалів, не втрачаючи жодної корисної інформації.

Модуль також управляє доступом до історії слів користувача, дозволяючи користувачам переглядати, редагувати та видаляти свої записи відповідно до своїх потреб. Ефективне управління історією слів забезпечує індивідуальний підхід до навчання, де кожен користувач може зосередитися на тих словах і виразах, які є найбільш релевантними для його процесу вивчення мови.

Взаємодія з користувацьким інтерфейсом забезпечує відображення та редагування збережених даних у зручному форматі, що робить процес навчання більш інтерактивним і персоналізованим. Користувач може легко знаходити потрібні записи через пошук або фільтри, а також отримувати додаткову інформацію про кожне слово чи речення, такі як приклади використання чи альтернативні переклади. Ці можливості сприяють кращому розумінню і запам’ятовуванню мовного матеріалу, роблячи модуль збереження даних важливою складовою ефективної системи вивчення іноземної мови.

Користувацький інтерфейс є центральним елементом, який забезпечує взаємодію користувача з системою, об'єднуючи всі функціональні компоненти і забезпечуючи зручний доступ до них. Він отримує введений текст та налаштування від користувача, які можуть включати вибір мови перекладу, стиль генерації речень,

рівень складності завдань та інші параметри. Ці дані негайно обробляються і передаються до відповідних модулів для подальшого опрацювання.

Після отримання введених даних користувача, інтерфейс передає їх до модуля генерації речень, який використовує GPT 3.5 для створення контекстуалізованих речень із зазначеними словами. Згенеровані речення передаються назад до інтерфейсу для виведення і перегляду. Одночасно інтерфейс передає дані до модуля перекладу, який використовує DeepL для перекладу згенерованих речень на обрану користувачем мову.

Коли модулі генерації речень та перекладу завершують свою роботу, результати негайно повертаються до користувацького інтерфейсу, який виводить їх на екран для перегляду користувачем. Це включає оригінальний текст разом з його перекладом, а також контекстуалізовані приклади у різних стилях та ситуаціях. Завдяки цьому користувач може легко порівнювати оригінальні та перекладені речення, а також аналізувати різні варіанти використання слів.

Користувацький інтерфейс діє як інтеграційний вузол, забезпечуючи безперервний і зручний потік інформації між користувачем і функціональними модулями системи. Це дозволяє користувачам легко вводити дані, отримувати результати перекладу та генерації речень, а також налаштовувати параметри системи відповідно до своїх навчальних потреб.

Модуль генерації речень є невід'ємним компонентом системи, який виконує функцію автоматичного створення контекстуалізованих прикладів речень. Він отримує ключові слова та контекст від користувача через інтерфейс, зокрема, слова або фрази, які користувач бажає вивчити, а також додаткові параметри, такі як стиль або специфічні ситуації для вживання. Ця інформація передається модулем через інтерактивний інтерфейс і слугує базою для подальшої обробки. Модуль генерації речень використовує API GPT 3.5 для створення прикладів речень, що відповідають заданим контекстуальним умовам. Обробка інформації відбувається в режимі реального часу, що дозволяє миттєво отримувати результати.

Після завершення процесу генерації, згенеровані речення передаються до модуля перекладу для подальшої обробки. Тут речення будуть перекладені на обрану користувачем мову за допомогою DeepL. Важливо зазначити, що передача

даних між модулями здійснюється швидко і безпечно, забезпечуючи інтегровану і плавну роботу системи. Завдяки таким інформаційним зв'язкам модуль генерації речень тісно взаємодіє з іншими компонентами системи, забезпечуючи користувача точними і корисними прикладами, що допомагає вивчати нову лексику ефективно і в контексті.

Модуль перекладу є критично важливим компонентом системи, який забезпечує високоякісний переклад згенерованих речень на обрану користувачем мову. Він отримує згенеровані речення від модуля генерації. Ці речення передаються до модуля перекладу через інтегровані інформаційні канали, забезпечуючи стабільний та швидкий потік даних для подальшої обробки.

Після завершення перекладу, модуль передає перекладені речення назад до користувацького інтерфейсу для відображення. Це дозволяє користувачам одразу побачити результати обробки, включаючи оригінальні речення та їхні переклади. Така інтеграція забезпечує безперервний цикл взаємодії, де користувач може легко перемикатися між оригінальними і перекладеними текстами, а також аналізувати різні варіанти використання слів і виразів у різних мовних контекстах. Завдяки чітким і надійним інформаційним зв'язкам між цими модулями, система забезпечує користувача якісними перекладами і покращує загальний досвід вивчення мови.

Для наглядності, побудована структурна схема розроблюваної системи (рисунок 2.1.1).

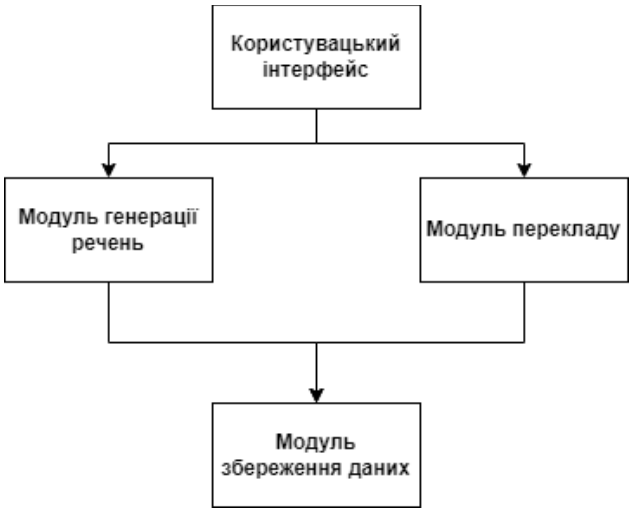


Рисунок 2.1.1 – Структурна схема системи

Представлена загальна структура проєктованої системи демонструє функціональні компоненти та їхні зв’язки, що дозволяють виконувати основні задачі модуля.

Разом діаграми класів (рисунок 2.1.2) і діаграми об'єктів (рисунок 2.1.3) формують статичну логічну модель системи, де класи визначають загальні шаблони для створення об'єктів, а діаграми об'єктів показують, як ці шаблони реалізуються і взаємодіють у реальних ситуаціях. Це дозволяє краще розуміти структуру системи, планувати її розвиток та забезпечувати ефективну взаємодію між різними її компонентами.

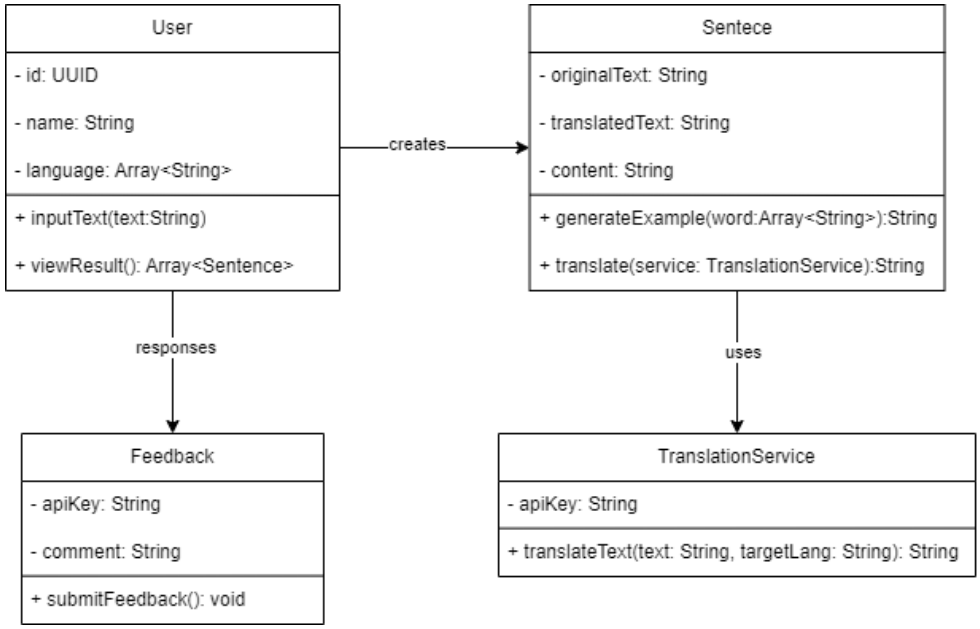


Рисунок 2.1.2 – Діаграми класів

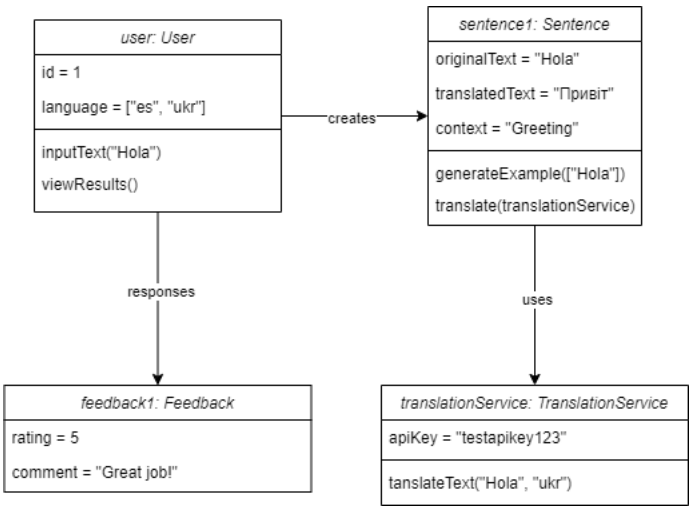


Рисунок 2.1.3 – Діаграми об’єктів

Елемент «User Interface» (див. рисунок 2.1.4) відображає інтерфейс користувача для введення тексту, перегляду результатів та взаємодії з іншими модулями.

«Sentence Module» Генерує контекстуалізовані речення, використовуючи обрані користувачем слова або фрази.

«Translation Module (DeepL)» Виконує переклад згенерованих речень, взаємодіючи з зовнішнім API DeepL.

«Feedback Module» Збирає зворотний зв'язок від користувачів для оцінки якості перекладу та генерації речень.

«Data Storage» Зберігає введені користувачем слова, згенеровані та перекладені речення, а також історію слів і фраз.

«Personalization & Adaptation Module» Аналізує прогрес користувача та надає персоналізовані рекомендації і адаптовані завдання.

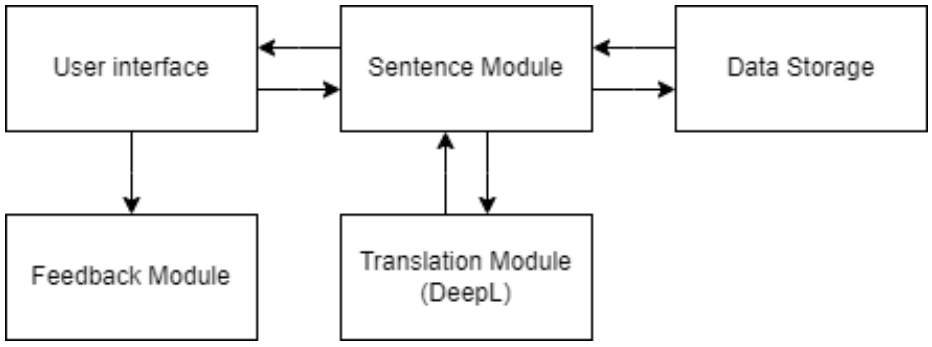


Рисунок 2.1.4 – Діаграми модулів

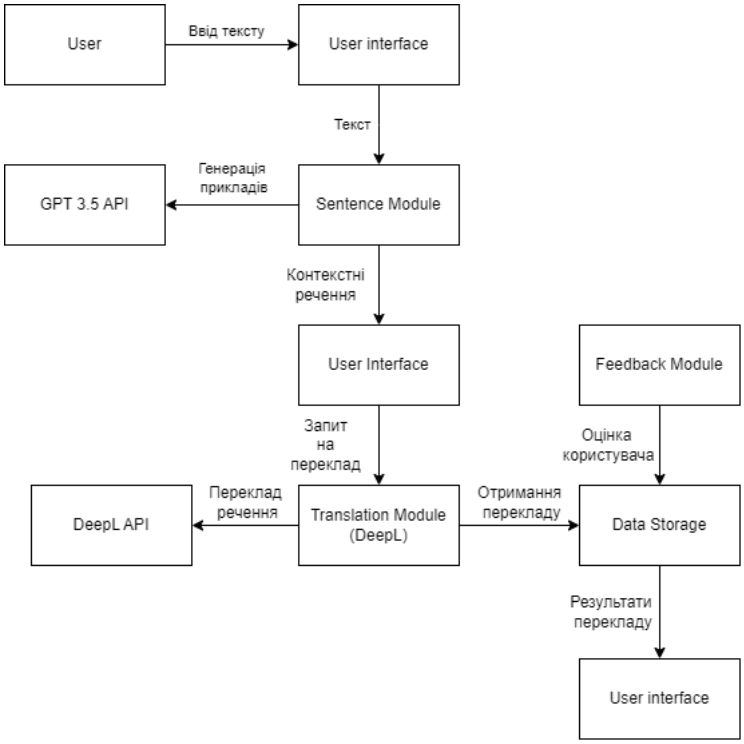


Рисунок 2.1.5 – Діаграми процесів

Діаграми модулів забезпечують чітке уявлення про організацію програмних компонентів і їх взаємозв'язки в системі, тоді як діаграми процесів описують, як дані передаються і обробляються між цими компонентами. Ці діаграми є важливою частиною статичної фізичної моделі, яка допомагає у плануванні, розробці та підтримці системи.

Діаграма переходів станів (рисунок 2.1.6) моделює процес вивчення іноземної лексики в програмному модулі, включаючи етапи вибору мови, додавання лексем, генерації контексту та перекладу, а також збереження результатів.

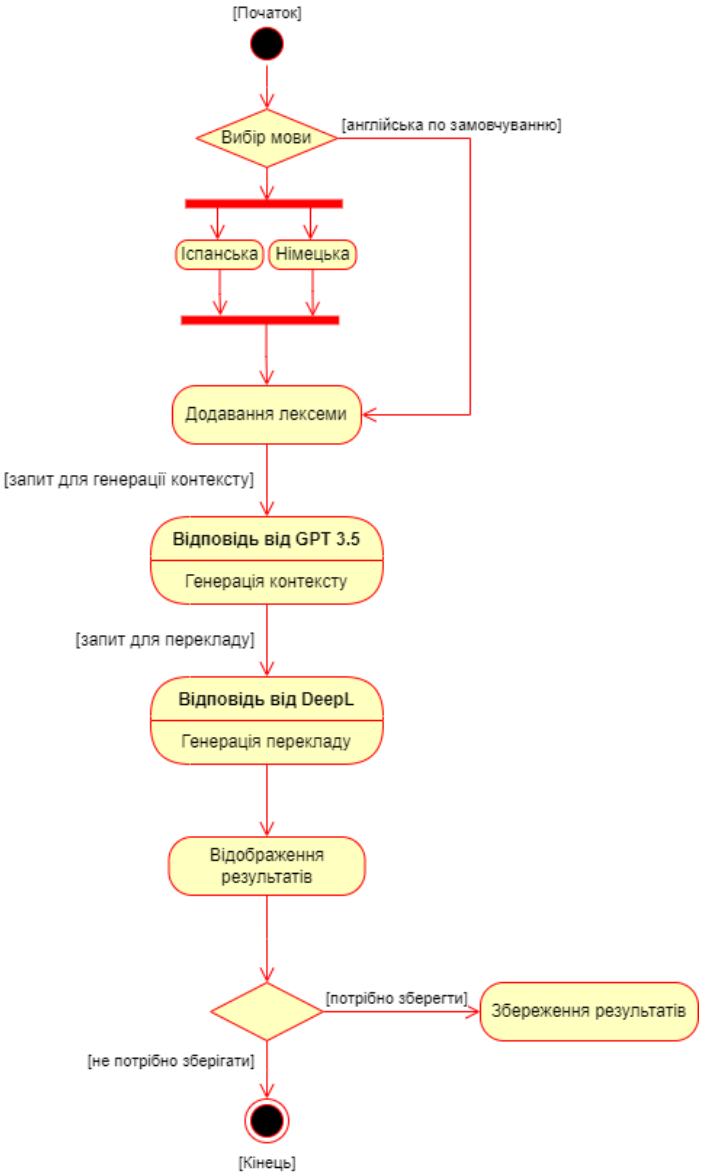


Рисунок 2.1.6 – Діаграми переходів станів

Ця діаграма чітко показує послідовність дій, які здійснює користувач та система, для того щоб забезпечити ефективне вивчення нових слів у контексті. Від вибору мови та додавання нової лексеми до обробки контексту і перекладу з використанням GPT-3.5 та DeepL, процес завершується самим збереженням або його відхиленням.

Часові системні діаграми, представлені на зображенні (рисунок 2.1.7), моделюють порядок взаємодій між користувачем, інтерфейсом користувача (UI), модулем обробки запитів (Request Processor), сервісом GPT-3.5, сервісом DeepL та базою даних SwiftData.

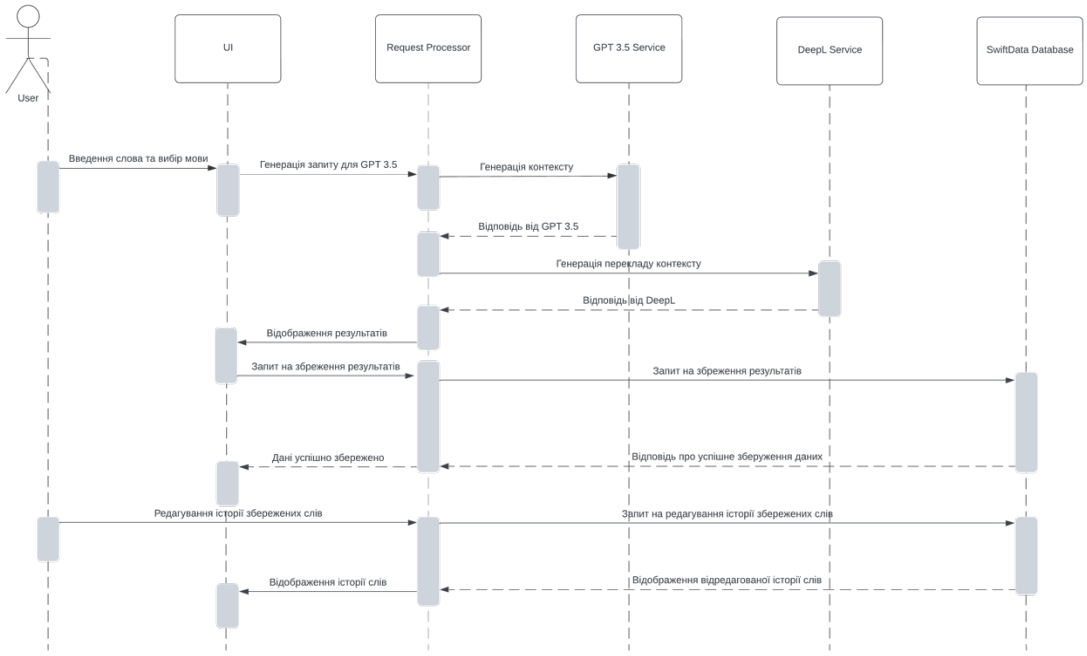


Рисунок 2.1.7 – Часові системні діаграми

Спочатку користувач вводить слово та вибирає мову в інтерфейсі користувача. Далі інтерфейс надсилає запит на генерацію контексту до модуля обробки запитів, який направляє запит до GPT-3.5 сервісу. Після отримання відповіді з контекстом, модуль обробки запитів надсилає контекст на переклад до DeepL сервісу. Отримавши переклад від DeepL сервісу, модуль обробки запитів відправляє результати назад до інтерфейсу користувача для відображення.

Якщо користувач вирішує зберегти результати, інтерфейс користувача надсилає відповідний запит на збереження до модуля обробки запитів, який передає дані до бази даних SwiftData. Після успішного збереження даних SwiftData надсилає підтвердження назад до модуля обробки запитів, який інформує інтерфейс користувача.

Користувач може редагувати історію збережених слів. Для цього інтерфейс користувача запитує редакцію історії збережених слів від бази даних через модуль обробки запитів, після чого відображає відредаговану історію слів користувачеві.

Діаграма демонструючи весь процес з введення, обробки та збереження нової лексеми та її контексту через послідовність повідомлень між різними компонентами системи, робить весь процес наочним та зрозумілим.

2.2 Алгоритмічне забезпечення

Алгоритм функціонування системи вивчення іноземної лексики починається з введення користувачем нового слова та вибору мови. Після цього система здійснює кілька кроків для генерації контексту, перекладу, відображення результатів, збереження отриманих даних та можливого редагування історії збережених слів.

Кожен блок схеми (додаток В) відповідає за певну логіку процесу і взаємодіє із зовнішніми службами та базами даних. Під час виконання алгоритму система переходить між блоками на основі результатів виконання відповідних запитів, враховуючи потенційні помилки та обробляючи їх відповідним чином для гарантії точності результатів. Алгоритм передбачає обробку потенційних помилок генерації контексту GPT-3.5 сервісом та помилок отримання перекладу DeepL сервісом. У цих випадках користувачеві буде повідомлено про помилку, і процес завершиться, щоб запобігти подальшим проблемам. Це забезпечує високу надійність системи і дозволяє швидко реагувати на будь-які несправності, гарантує коректність опрацювання даних та формування результатів.

2.3 Інформаційне забезпечення

2.3.1 Опис вхідної та вихідної інформації

Варто представити детальний опис вхідної та вихідної інформації для кожної основної функції системи. До вхідної інформації функції вибору мови та введення нового слова відноситься сама мова, або вибір користувачем мови для вивчення (наприклад, іспанська, німецька, англійська). Також об’єктом вхідної інформації є слово (лексема), введене користувачем для детальнішого вивчення контексту його вживання. До вихідної інформації належить підтвердження введення, це підтвердження інтерфейсом користувача, що нове слово та вибрана мова збережені для подальшої обробки. Якщо описувати функцію генерації контексту, то до вхідної інформації належать так само, мова та слово, проте тепер на виході буде контекст, який був згенерований сервісом GPT-3.5 для введеного слова. До генерації перекладу на вхід подається згенерований контекст слова та мова перекладу, на

виході отримується перекладений текст контексту, наданий сервісом DeepL. Для відображення результатів надходять контент та переклад, після чого на виході в інтерфейс відображається згенерований контекст і відповідний переклад.

2.3.2 Концептуальне інфологічне проєктування

Глобальна інфологічна модель даних (рисунок 2.3.2.1) описує загальну структуру даних для всієї системи і охоплює всі сутності та зв'язки між ними на високому рівні.

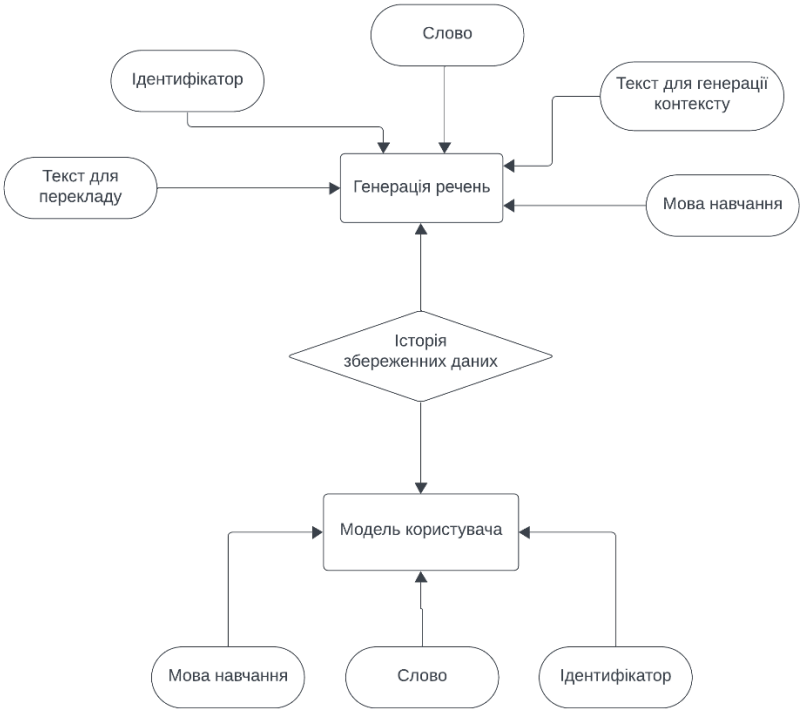


Рисунок 2.3.2.1 – Глобальна інфологічна модель

Локальна інфологічна модель даних (рисунок 2.3.2.2) детальніше розглядає підсистеми або окремі компоненти системи, деталізуючи інформацію для кожної з сутностей і показуючи внутрішні взаємодії в контексті окремої області або підсистеми.

Рисунок 2.3.2.2 – Локальна інфологічна модель

Концептуальне інфологічне проектування забезпечує чітке визначення структури даних і взаємозв'язків між ними, що є важливим кроком для подальшого розроблення програмного модуля.

2.3.3 Проєктування глобальної даталогічної моделі даних.

ER-модель (рисунок 2.3.3.1) як зазначено у працях [12, 13] це модель даних, яка описує концептуальні схеми предметної області вивчення іноземної мови на основі штучного інтелекту.

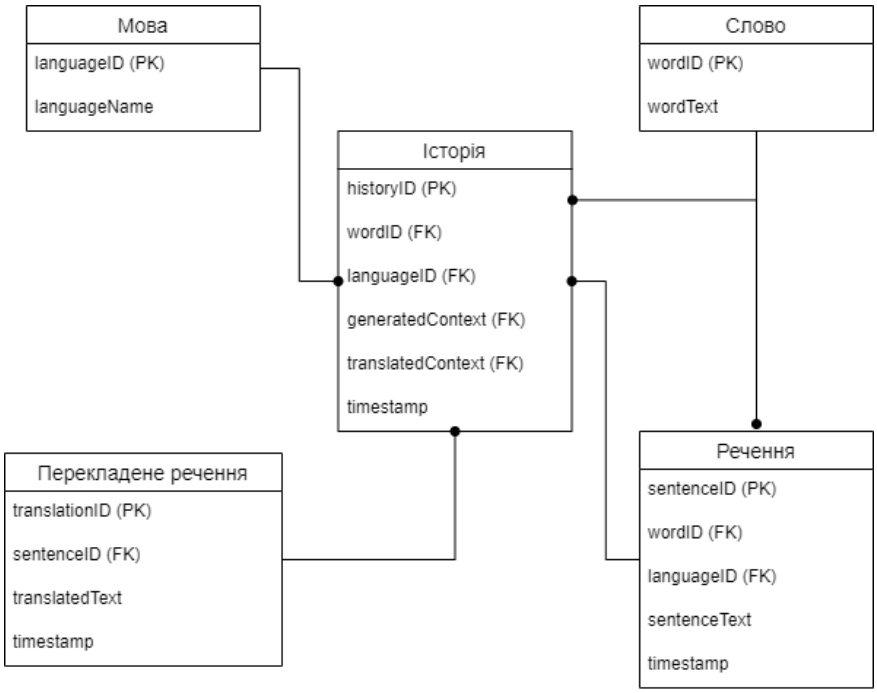


Рисунок 2.3.3.1 – Модель даних у вигляді ERD (у нотації IDEF1X)

Реляційна модель (рисунок 2.3.3.2), як зазначено в роботі [10], дозволяє визначити структуру бази даних і зв’язки між таблицями. Враховуючи вимоги щодо генерації контекстуалізованих речень і їх перекладу.

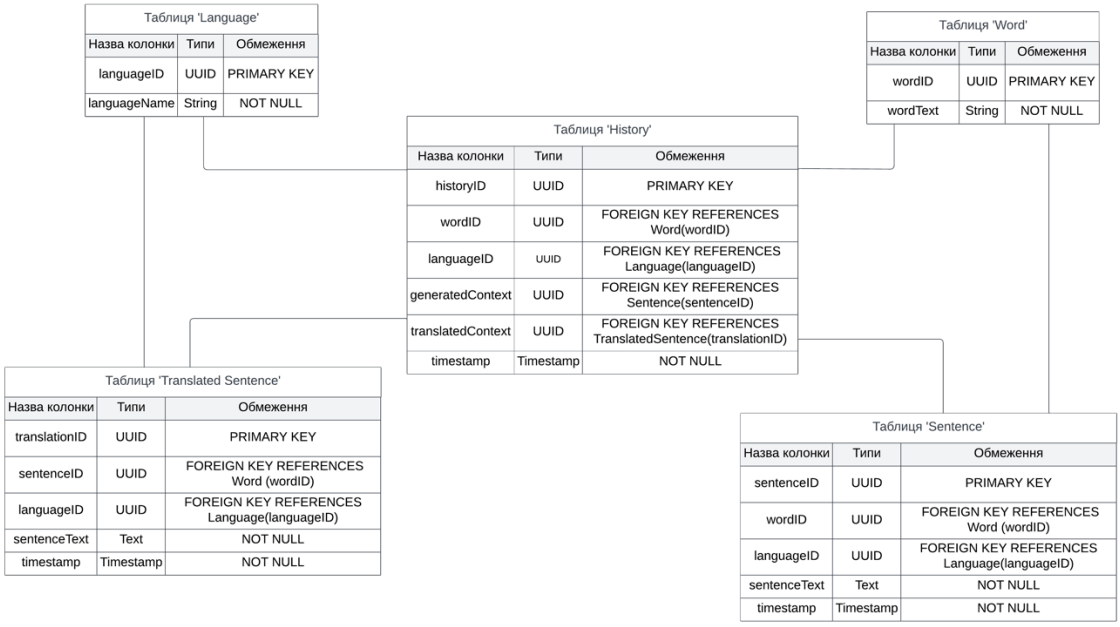


Рисунок 2.3.3.2 – Реляційна модель

Ця реляційна модель структурує дані таким чином, що з одного введеного слова генерується контекстуалізоване речення, яке потім перекладається, і вся інформація зберігається для подальшого аналізу та використання.

2.3.4 Проєктування фізичної моделі даних

Проєктування фізичної моделі (рисунок 2.3.4.1) даних, як згадано у праці [13] включає детальний опис структури бази даних, її таблиць та атрибутів.

UUID: Використовується для унікальних ідентифікаторів, щоб уникнути зіткнень, особливо в масштабованих системах. VARCHAR: Забезпечує зберігання текстових даних з обмеженням до кількості символів вказаних в дужках. TEXT: Використовується для зберігання великого обсягу текстових даних. TIMESTAMP: Використовується для зберігання дати і часу.

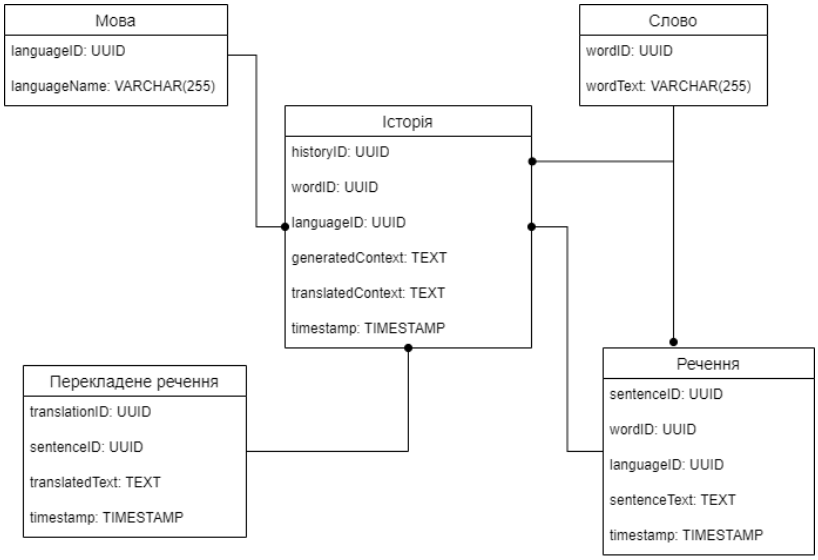


Рисунок 2.3.4.1 – Фізична модель даних

2.3.5 Програмна реалізація бази даних

Програмна реалізація бази даних (додаток А.2), написана на Swift із використанням SwiftData, містить модель даних `ModelForWord`, яка представляє слова, їх переклад та контекстуальні речення.

Спочатку імпортується бібліотека SwiftData за допомогою `import SwiftData`, що забезпечує необхідні інструменти для роботи з базою даних.

Клас `ModelForWord` позначено декоратором `@Model`, що вказує на його використання як моделі даних у контексті бази даних SwiftData. Клас містить п'ять властивостей: `name`, `responce`, `translate`, `creationDate`, та `language`.

`name`: Строкове поле, яке містить слово або назву, яке користувач вводить для генерації та перекладу контекстуальних речень.

`responce`: Масив строк, що містить контекстуалізовані речення від сервісу, наприклад GPT-3.5, для заданого слова.

`translate`: Масив строк, що містить переклади контекстуалізованих речень за допомогою сервісу, наприклад DeepL.

`creationDate`: Поле типу `Date`, яке автоматично встановлюється на поточний час під час створення нового об'єкта моделі. Це дозволяє фіксувати час створення запису.

`language`: Строкове поле, яке визначає мову для введеного слова (наприклад, «English», «Spanish» тощо).

Клас також має ініціалізатор, який приймає параметри для ініціалізації всіх властивостей, за виключенням `creationDate`, яка автоматично встановлюється на поточний час за допомогою `.now`.

В результаті, модель `ModelForWord` зберігає інформацію про слова, згенерований контекст та переклади, що дозволяє ефективно керувати цими даними в базі даних SwiftData.

3 ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ (МОДУЛЯ)

3.1 Реалізація програмного забезпечення

Структурна схема (рисунок 3.1.1) програмної системи графічно відображає компоненти системи, їхні зв'язки та взаємодію. В схемі описується взаємодія між різними компонентами системи, включаючи користувача, інтерфейс користувача, контролер, сервіси GPT 3.5 і DeepL, а також базу даних. Така схема полегшує розуміння потоків даних і функціональних зв’язків між компонентами програмної системи.

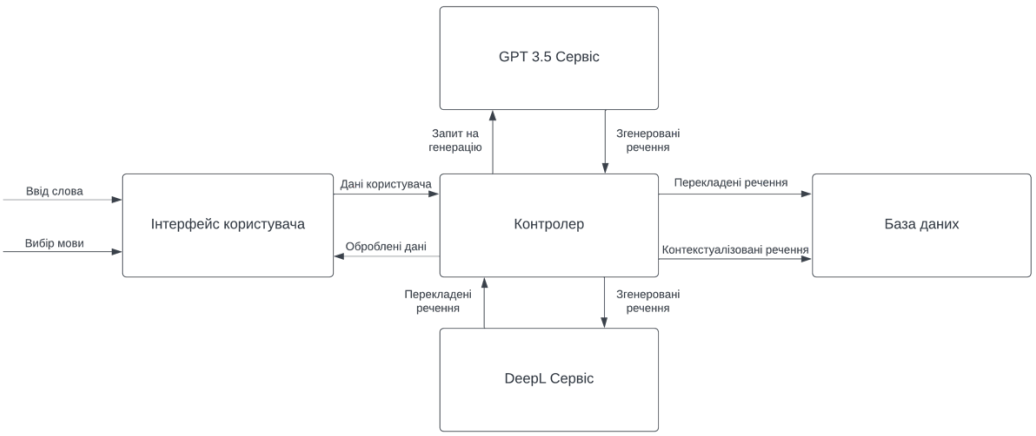


Рисунок 3.1.1 – Структурна схема програмної системи

UML-діаграма класів (додаток С), як згадано в праці [19], реалізує основну бізнес-логіку програмної системи.

Клас «Мова» відповідає за зберігання інформації про мови. Він містить унікальний ідентифікатор «languageID» та назву мови «languageName». До методів відноситься конструктор для ініціалізації параметрів мови.

Клас «Слово» зберігає слова, введені користувачем. Він містить унікальний ідентифікатор «wordID» та текст слова «wordText». Його методами є конструктор для ініціалізації параметрів слова та метод для збереження нового слова до бази даних.

«Речення» це клас який зберігає контекстуалізовані речення для заданого слова. Він містить унікальний ідентифікатор «sentenceID», текст речення «sentenceText», зовнішні ключі «wordID» та «languageID», а також «timestamp» для

відстеження часу створення речення. До методів входить конструктор для ініціалізації параметрів речення та метод для збереження згенерованого речення до бази даних.

Клас «Перекладене речення» зберігає перекладені речення. Він містить унікальний ідентифікатор «translationID», зовнішній ключ «sentenceID», перекладений текст «translatedText», а також «timestamp» для відстеження часу створення перекладу. До його методів входить конструктор для ініціалізації параметрів перекладеного речення та метод для збереження перекладеного речення до бази даних.

Клас «Історія» зберігає історію запитів. Він містить унікальний ідентифікатор «historyID», зовнішні ключі «wordID» та «languageID», масиви ідентифікаторів згенерованих контекстів «generatedContext» та перекладів «translatedContext», а також «timestamp» для відстеження часу створення запису історії. До методів відноситься конструктор для ініціалізації параметрів історії та метод для збереження історії до бази даних.

UML-діаграма станів (рисунок 3.1.3) показує різні стани, в яких можуть знаходитися елементи графічного інтерфейсу користувача, і переходи між цими станами внаслідок певних дій користувача або системних подій, як зазначено у праці [21].

У стані введення слова та вибору мови, користувач вводить дані. Запит на генерацію це система, яка відправляє запит до GPT 3.5 для генерації контексту. Згенерований контекст відповідає за підтвердження успішної генерації контексту. Переклад контексту виконує запит до DeepL для перекладу контекстуальних речень. Відображення результатів показує перекладені речення користувачу. Стан збереження даних у історію відповідальний за збереження результатів запитів до бази даних та відображення їх у історії коистувача. У повідомленні про помилку система інформує користувача про помилки зв’язані з сервісами. Кінець, це логічне завершення операції.

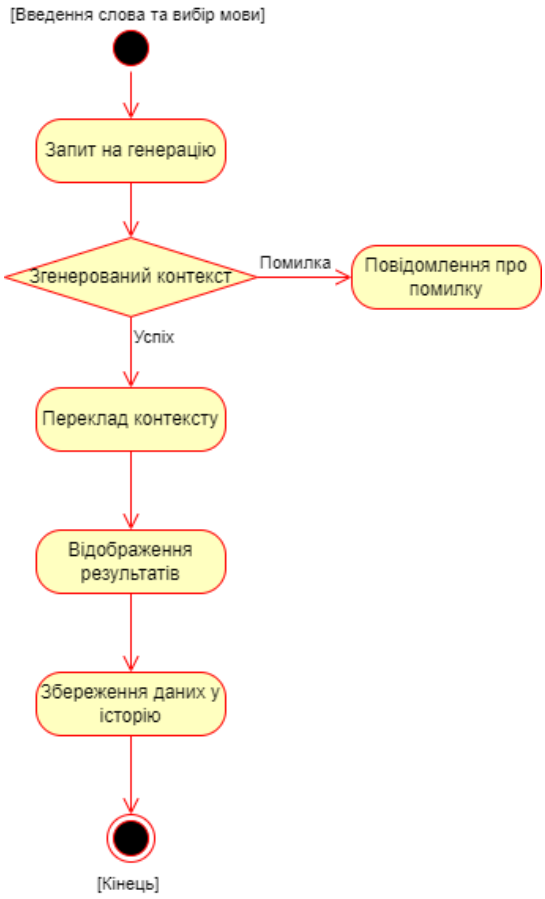


Рисунок 3.1.3 – UML-діаграма станів

Обґрунтування вибору програмних засобів реалізації

Для реалізації програмної системи було обрано використання мови програмування Swift, фреймворку SwiftUI та бібліотеки SwiftData, оскільки вони забезпечують ефективне та зручне розроблення сучасних iOS та macOS додатків. Swift, як мова програмування, є сучасною, безпечною та продуктивною, створеною спеціально для екосистеми Apple. Це забезпечує високу продуктивність додатку та оптимізовану роботу з ресурсами пристрою. SwiftUI, це фреймворк для побудови користувацьких інтерфейсів він дозволяє розробляти інтуїтивно зрозумілі та реактивні інтерфейси з мінімальним кодом. Завдяки декларативному підходу SwiftUI спрощує розробку та управління станами інтерфейсу, покращуючи тим самим загальну продуктивність розробки та зручність підтримки. Використання бібліотеки SwiftData надає зручний інтерфейс для роботи з локальною базою даних, зберігання та доступ до даних додатку, що в поєднанні з можливостями Swift і SwiftUI дозволяє легко інтегрувати збереження даних у додатку. Таким чином, обрані технології забезпечують високу продуктивність, зручність розробки та

інтеграцію з екосистемою Apple і підтримку сучасних практик розробки додатків, що є критично важливими для створення ефективної та надійної програмної системи.

Лістинг бізнес логіки виконаний у Додатку А.

Для генерації користувач вводить слово, яке необхідно вивчити, в текстове поле. При натисканні кнопки, викликається функція `gptResponse()`, яка формує запит до GPT 3.5 API. Запит містить інструкції, що генерувати три речення з використанням вказаного слова в різних контекстах. Після цього, відповідь API парсується для отримання окремих речень.

Після генерації іншомовних речень, функція `deepTranslateText()` автоматично викликає DeepL API для перекладу згенерованих речень на обрану мову (наприклад, іспанську або німецьку). Перекладені речення формуються в масив та відображаються користувачу.

Для збереження історії запитів та відповідей використовується SwiftData. Кожен запит та відповідні згенеровані і перекладені речення зберігаються у локальній базі даних. Це дозволяє користувачу отримувати доступ до попередніх запитів і їх результатів через інтерфейс програми.

3.2 Інтерфейс користувача

Головна сторінка додатку (рисунок 3.2.1) дозволяє користувачам вводити слово для отримання прикладів речень та їх перекладів. Вона має інтуїтивний і простий дизайн, що складається з кількох ключових елементів. У верхній частині екрану знаходиться заголовок «Sentences», який відображає контекст поточного сеансу. Поруч з заголовком розташовано дві кнопки, одна з яких дозволяє розгорнути або згорнути додаткові переклади, а інша веде до історії попередніх запитів. У центрі сторінки розташоване повідомлення «Введіть слово», яке спонукає користувача до дії. Під ним, у нижній частині екрану, є інтерфейс для введення слова - текстове поле з написом «Send message» та кнопка для надсилання повідомлення. Ліва частина цього інтерфейсу містить іконку з прапором, яка відображає обрану мову перекладу і відкриває меню для вибору інших мов. Загальний дизайн сторінки є мінімалістичним і орієнтованим на зручність

користувача, забезпечуючи швидкий доступ до основних функцій додатку. В інтерфейсі реалізовано анімації для плавного розкриття та закриття додаткової інформації про перекладені речення. Користувач може розгорнути або згорнути перекладені речення, натиснувши відповідну кнопку. Також використовується анімація для плавного відображення результатів, коли вони стають доступними.

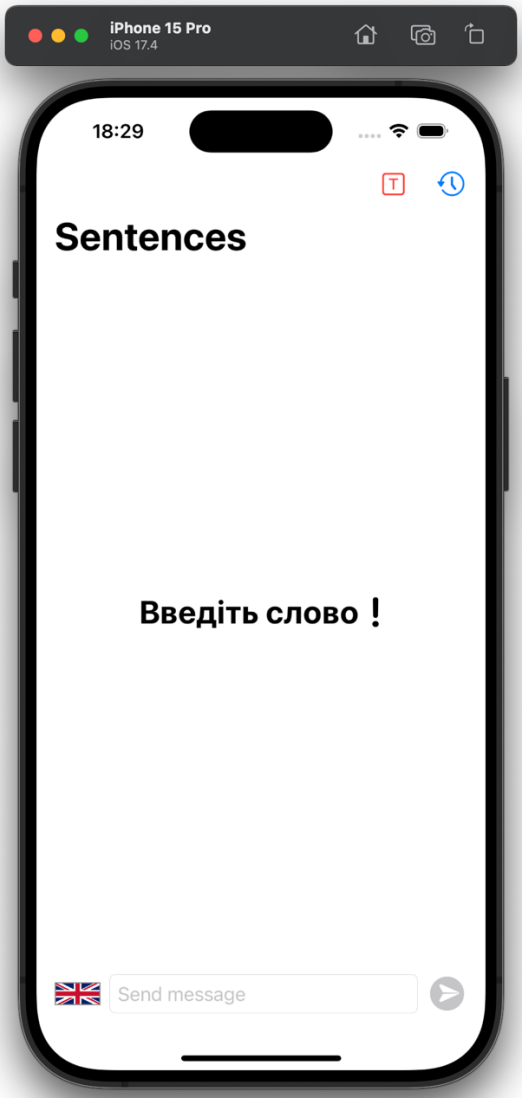


Рисунок 3.2.1 – Інтерфейс головної сторінки програмного модуля

На сторінці «History» програмного модуля (рисунок 3.2.2) користувачу надається доступ до журналу раніше виконаних запитів щодо генерації речень і їх перекладів. У верхній частині екрану знаходиться заголовок «History», що показує зміст цієї сторінки. Поруч із заголовком знаходиться кнопка для видалення записів з історії, що може бути корисно для користувача, який хоче очистити журнал

запитів. Під заголовком розташований перелік запитів у вигляді карток, кожна з яких містить результати згенерованих речень для заданого слова.

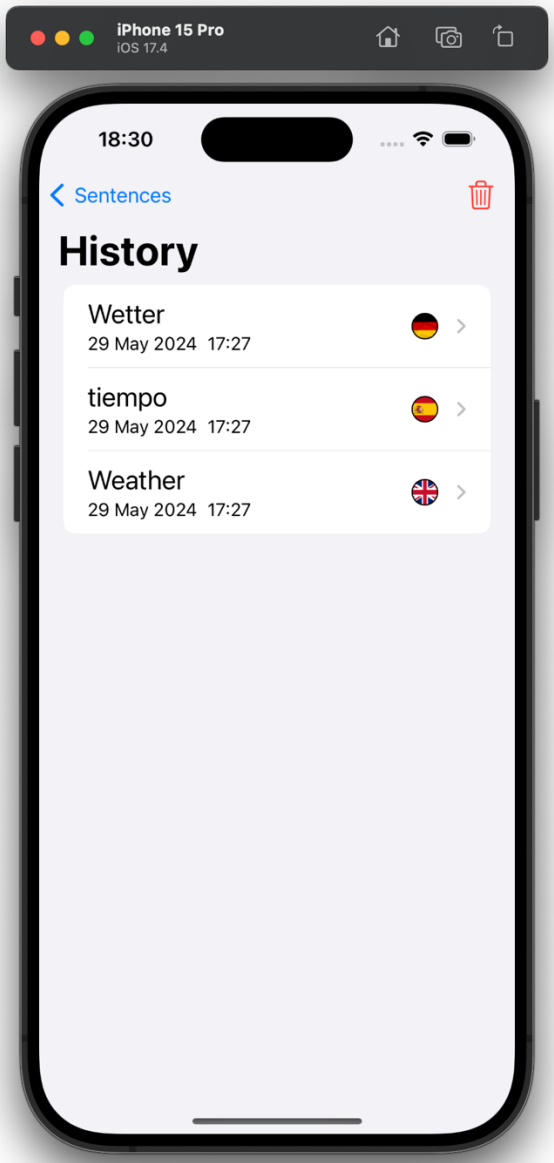


Рисунок 3.2.2 – Інтерфейс сторінки історії запитів програмного модуля

Вкладка виконаних запитів містить слово, для якого проводився запит, дату та час запиту, що дозволяє користувачу легко орієнтуватися у хронології своїх дій та іконка прапора, яка вказує на мову, на яку було перекладено згенеровані речення

Крім того, права сторона кожної картки містить стрілку, що дозволяє користувачу натиснути на неї для отримання детальної інформації щодо конкретного запиту. Загалом, сторінка «History» забезпечує зручний доступ до збереженої інформації, дозволяючи користувачу повторно переглядати раніше

згенеровані та перекладені речення, тим самим підвищуючи ефективність навчання та утримання лексики.

3.3 Тестування розробленої програми

Тестування розробленої програми є важливим етапом, який забезпечує її стабільність, функціональність та зручність у використанні. Для цього використовуються різні підходи та інструменти, включаючи автоматизоване тестування, ручне тестування та симуляцію середовища розробки в Xcode.

Автоматизоване тестування з використанням XCTest – стандартного фреймворку для тестування в середовищі розробки Xcode. В цьому процесі створюються тестові кейси, які перевіряють основні функціональні можливості програми, такі як генерація речень за допомогою GPT 3.5 API, переклад речень з використанням DeepL API, правильність відображення результатів, а також збереження та відображення історії запитів. Ці тести автоматично запускаються після кожної зміни в коді, забезпечуючи негайний зворотний зв'язок з розробниками щодо можливих помилок або невідповідностей.

Ручне тестування (рисунок 3.3.1) є невід'ємною частиною процесу. Воно включає перевірку користувацьких запитів, починаючи від введення слова до отримання згенерованих речень і їх перекладу, а також взаємодію з історією запитів. Процес тестування (додаток В.3) здійснюється за допомогою середовища Xcode на різних пристроях з різними версіями iOS, це забезпечує сумісність та стабільність додатку на всіх підтримуваних платформах.

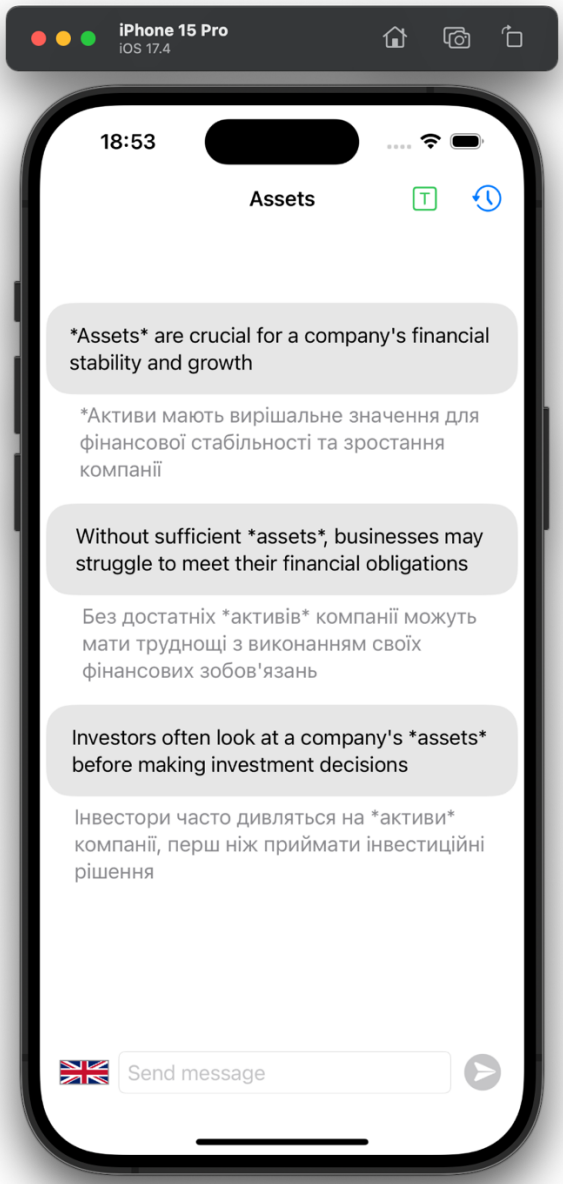


Рисунок 3.3.1 – Процес ручного тестування програмного модуля

Особлива увага приділяється тестуванню коректного функціонування мережевих запитів до API сервісів GPT та DeepL. Перевіряється, чи правильно формуються запити, обробляються відповіді та відображаються повідомлення про помилки у разі їх виникнення.

Ще одним важливим аспектом тестування є перевірка продуктивності та ресурсоємності додатку. Використовуючи інструменти профілювання в Xcode, такі як Instruments, аналізується використання оперативної пам'яті, процесорний час, а також швидкість виконання основних операцій. Це дозволяє виявити та

оптимізувати найбільш критичні місця в коді, забезпечуючи швидкість та плавність роботи додатку.

Завдяки комплексному підходу до тестування, який включає автоматизовані тести, ручні перевірки та профілювання продуктивності, присутня впевненість у надійності, функціональності та зручності програмного модуля для користувача.

ВИСНОВКИ

Ця робота включала ретельний аналіз існуючих методів вивчення іноземних мов, вибір найбільш ефективних підходів, інтеграцію сучасних мікро-сервісів, таких як GPT 3.5 та DeepL API, для генерації і перекладу речень, а також розробку користувацького інтерфейсу з використанням SwiftUI.

Одним з головних досягнень цієї роботи є створення інтуїтивного користувацького інтерфейсу, який дозволяє ефективно взаємодіяти з додатком. Інтерфейс забезпечує зручний процес введення слів, отримання контекстуалізованих речень та їх перекладів, а також зберігання історії запитів для подальшого використання. Використання SwiftUI дозволило створити легко масштабовану та підтримувану архітектуру інтерфейсу, яка славиться своєю гнучкістю та простотою у підтримці.

Інтеграція GPT 3.5 API значно підвищила функціональні можливості додатку, дозволивши автоматично генерувати контекстуалізовані речення для введених користувачем слів. Це забезпечує глибше розуміння значення та використання слова у різних контекстах. Використання DeepL API для перекладу цих речень на різні мови робить програмний модуль універсальним інструментом для вивчення кількох мов за допомогою одного додатку.

Особливу увагу в роботі було приділено тестуванню розробленої програми. Застосування методів автоматизованого та ручного тестування забезпечило надійність та стабільність роботи програмного модуля. Тестування включало в себе перевірку коректної роботи основних функцій, виробничу перевірку мережевих запитів до API сервісів, аналіз використання ресурсів та продуктивності, а також тестування на різних пристроях і версіях операційної системи iOS. Ці заходи дозволили своєчасно виявити і виправити недоліки, що сприяло підвищенню якості кінцевого продукту.

Таким чином, результатом проведених досліджень і розробок став програмний модуль, здатний значно полегшити процес вивчення іноземних мов, зокрема засвоєння нової лексики через контекстуалізоване використання слів. Завдяки інтеграції сучасних мікро-сервісів і створенню зручного інтерфейсу, ця

програма може бути успішно використана як індивідуальними учнями, так і в освітніх установах для навчання мовам. Ця робота є вагомим внеском у галузь розробки програмних засобів для навчання і надає нові можливості для покращення методик вивчення іноземних мов.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Brown, H. D. Principles of Language Learning and Teaching. Pearson Longman, 2007. 410p (ст.9 - 10)

2. Johnson, K. An Introduction to Foreign Language Learning and Teaching. Pearson Education, 2008. 368p (ст.9 - 10)

3. Ellis, R. Second Language Acquisition. Oxford University Press, 1997. 400p (ст.14 –18)

4. Chapelle C. A. Computer Applications in Second Language Acquisition. Cambridge University Press, 2001. 221p (ст.14 – 18)

5. Krashen S. D. Principles and Practice in Second Language Acquisition. Pergamon Press, 1982. 202p (с.14 – 18)

6. Peterson M. Massively Multiplayer Online Role-Playing Games (MMORPGs) in Education. Interactive Technology and Smart Education, 2010. 189p (ст.19)

7. Kachru B. B. The Other Tongue: English across Cultures. University of Illinois Press, 1992. 356p (ст.29)

8. Harmer J. The Practice of English Language Teaching. Pearson Longman, 2007. 448p (ст. 9)

9. Nunan D. Second Language Teaching & Learning. Heinle & Heinle Publishers, 1999. 482p (ст. 20 – 21)

10. Halpin T., Morgan T. Information Modeling and Relational Databases. Morgan Kaufmann, 2008. 976p (ст. 29 – 31)

11. Batini C., Scannapieco M. Data and Information Quality: Dimensions, Principles and Techniques. Springer, 2016. 500p (ст. 29 – 32)

12. Hay D. C. Data Model Patterns: Conventions of Thought. Dorset House Publishing, 2006. 412p (ст. 30 – 34)

13. Simsion G. C. Data Modeling Essentials. Morgan Kaufmann, 2007. 560p (ст. 30 – 33)

14. Hacking with Swift (2021). 100 Days of SwiftUI. Retrieved from [Hacking with Swift](https://www.hackingwithswift.com/100/swiftui)

15. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. McGraw-Hill Education, 2014. 976p (ст. 19 – 29)

16. Richards M., Ford N. Fundamentals of Software Architecture: An Engineering Approach. O'Reilly Media, 2020. 432p (ст. 15 – 16)

17. Дідух Д. Р. «ПРОГРАМНИЙ МОДУЛЬ ДЛЯ ВИВЧЕННЯ ІНОЗЕМНОЇ МОВИ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ» прийняті до публікації в збірнику за матеріалами VI Всеукраїнської студентської наукової конференції «Експериментальні та теоретичні дослідження в контексті сучасної науки» (21.06.2024, м. Рівне, Україна).

18. Загальні методичні рекомендації з підготовки, оформлення, захисту та оцінювання кваліфікаційних робіт здобувачів вищої освіти першого (бакалаврського) і другого (магістерського) рівнів. Тернопіль: ЗУНУ, 2024. 83 с.

19. Larman C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Prentice Hall, 2004. 736p (ст. 35 – 38)

20. Fong A. C., Bi N. Web Services and Service-Oriented Computing. Springer, 2013. 269p (ст. 14 – 18)

21. Fowler M., Scott K. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional, 1999. 208p (ст. 35 – 38)

22. GPT-3: Language Models are Few-Shot Learners. URL: https://arxiv.org/pdf/2005.14165 (дата звернення: 12.03.2024)

23. Apple Inc. SwiftUI Documentation. URL: https://developer.apple.com/documentation/swiftui/ (дата звернення: 21.04.2024)

24. Apple Inc. Human Interface Guidelines. URL: https://developer.apple.com/design/human-interface-guidelines/ (дата звернення: 20.04.2024)

25. DeepL GmbH. DeepL API Documentation. URL: https://developers.deepl.com/docs (дата звернення: 15.04.2024)

Додаток А.1

Код основної бізнес-логіки програмного модуля

```
import SwiftUI
import SwiftData

struct ContentViev: View {
   @Environment (\.modelContext) var modelContext
   @Query(sort: \ModelForWord.creationDate, order: .reverse) var words: [ModelForWord]

   @State private var responseText = ""
   @State private var responseArray: [String] = []
   @State private var DeepResponseText = ""
   @State private var DeepResponseArray: [String] = []
   @State private var customPrompt = ""
   @State private var responsePrompt: String = ""
   @State private var label = ""
   @State private var labelStatus = false
   @State private var showingStatus = false
   @State private var isExpanded: Bool = false
   @State private var langSelection = "English"
   @State private var language = ["English", "Espanol", "Deutsch"]
   @State private var prompt = ""

   let DeepApiKey = "9ccb63ac-dd87-4d7f-b09a-ea213ebc61ce:fx"
   let history = Image(systemName: "history")
   let apiKey = "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"
   var body: some View {
      NavigationStack{
         VStack{

            if showingStatus{
               if DeepResponseArray.isEmpty{
                  Spacer()
                  Loader()
                  Spacer()
               }else{

                  Spacer()

                  VStack {
                     ForEach(0..<3, id: \.self) { i in
                        VStack {
                           Text("\(self.responseArray[i].trimmingCharacters(in: .whitespacesAndNewlines))")
                              .padding()
                              .frame(maxWidth: .infinity)
                              .background(Color.primary.opacity(0.1))
                              .clipShape(RoundedRectangle(cornerRadius: 20))

                           if isExpanded {
                              Text("\(self.DeepResponseArray[i].trimmingCharacters(in: .whitespacesAndNewlines))")
                                 .foregroundColor(.secondary)
                                 .padding(.horizontal)
                           }
                        }
                        .padding(5)
                        .animation(.spring(), value: isExpanded)
                     }
                  }

                  .onAppear {

                  }
```

```
            Spacer()
        }
    }else{
        Spacer()
        HStack{
            Text("Введіть слово")
                .font(.title)
                .bold()
            Image(systemName: "exclamationmark")
                .font(.title)
                .bold()
        }

    }
    Spacer()
    HStack{
        Menu{
        Picker("Select language",selection: $langSelection){
            ForEach(0..<language.count){ (i) in
                HStack{
                    Text(self.language[i])
                    Image(self.language[i])
                        .scaledToFit()

                }.tag(self.language[i])
            }
        }
    }label: {
            Image(langSelection)
            .resizable()
            .scaledToFit()
            .frame(width: 40)
            .border(.gray)

        }

        TextField("Send message", text: $customPrompt)
            .textFieldStyle(.roundedBorder)
            .keyboardType(.alphabet)


        Button {
            withAnimation{
                print(customPrompt)
                showingStatus = true
                labelStatus = true
                label = customPrompt
                DeepResponseArray = []
                gptResponse()
                customPrompt = ""
            }
        } label: {
            Image(systemName: "paperplane.circle.fill")
                .rotationEffect(.degrees(45))
                .font(.system(size: 30))
        }
        .disabled(customPrompt.count < 2)
    }
    .padding()
}
.navigationTitle(labelStatus ? label : "Sentences")
.navigationBarTitleDisplayMode(labelStatus ? .inline : .automatic)
.toolbar{
    ToolbarItemGroup(placement: .topBarTrailing){
```

```
                Button{
                    withAnimation{
                        isExpanded.toggle()
                    }
                }label: {
                    Image(systemName: "t.square")
                        .foregroundStyle(isExpanded ? .green : .red)
                }

                NavigationLink(destination: GPTUIView(words: words)){
                    Image(systemName: "clock.arrow.circlepath")
                }



            }
        }
    }
}
func addword(){
        let example = ModelForWord(name: label, responce: responseArray, translate: DeepResponseArray, language:
langSelection)
        modelContext.insert(example)

}
func deepTranslateText() {
    translateText(text: responseText, apiKey: DeepApiKey, lang: langSelection) { result in
        switch result {
        case .success(let translatedText):
            DeepResponseText = translatedText
            DeepResponseArray = DeepResponseText.components(separatedBy: ".")
            print(DeepResponseArray)
        case .failure(let error):
            DeepResponseText = "Error: \(error.localizedDescription)"
        }
    }
    sleep(3)
    addword()
}

func gptResponse() {
    prompt = "Create three sentences in \(langSelection) with different values for the word \(customPrompt) (replace with
the actual word). Start each sentence from a new paragraph. Emphasize the key word in each sentence by surrounding it
with asterisks\(customPrompt). The sentences should be in the format '[sentence]'. For example: Sentence 1 - This is a
sentence with \(customPrompt). Sentence 2 - Another sentence with a different \(customPrompt). Sentence 3 - Yet another
sentence with yet another \(customPrompt). Please note that the actual sentences should have different values for the word
'\(customPrompt)', and the key word in each sentence should be \(customPrompt). This revised prompt clarifies the
requirements for ChatGPT, ensuring that it generates the correct output in the desired format."
    getChatGPTResponse(prompt: prompt, apiKey: apiKey) { result in
        switch result {
        case .success(let response):
            responseText = response
            responseArray = responseText.components(separatedBy: ".")
            deepTranslateText()
        case .failure(let error):
            responseText = "Error: \(error.localizedDescription)"
        }
    }

}
}
```

Додаток А.2

Код реалізації моделей бази даних у програмному модулі

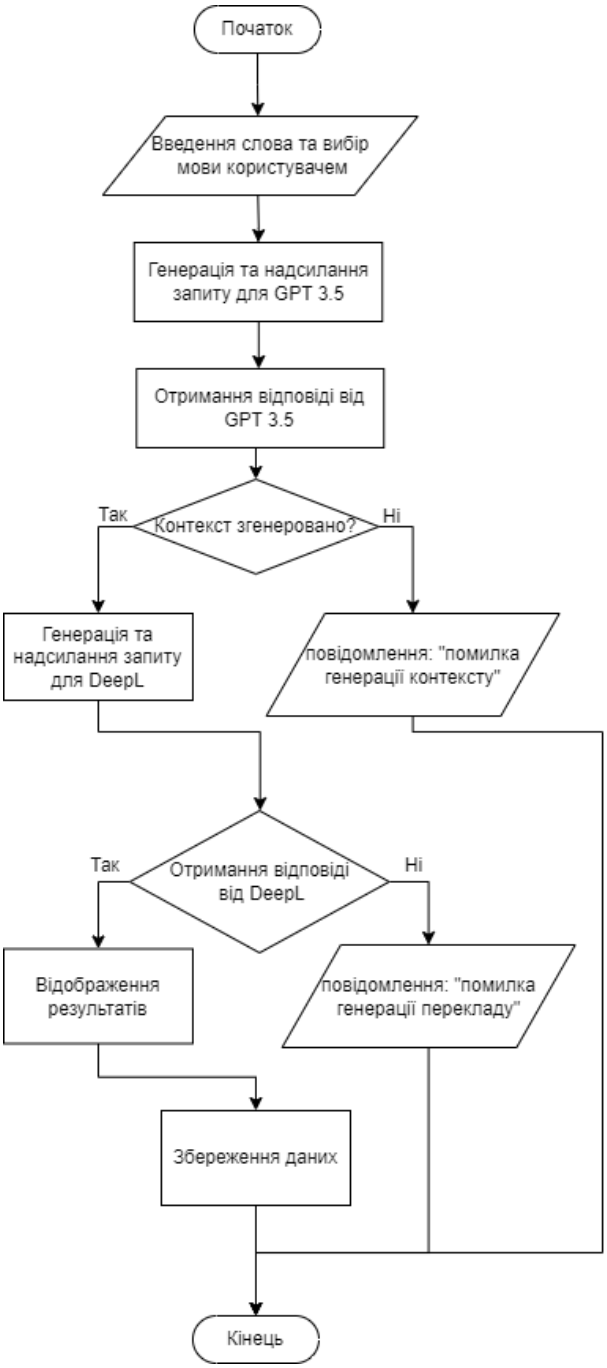
```
import Foundation
import SwiftData

@Model
class ModelForWord {
    var name: String
    var responce: [String]
    var translate: [String]
    var creationDate: Date
    var language: String

    init(name: String, responce: [String], translate: [String], language: String) {
        self.name = name
        self.responce = responce
        self.translate = translate
        self.creationDate = .now
        self.language = language
    }
}
```

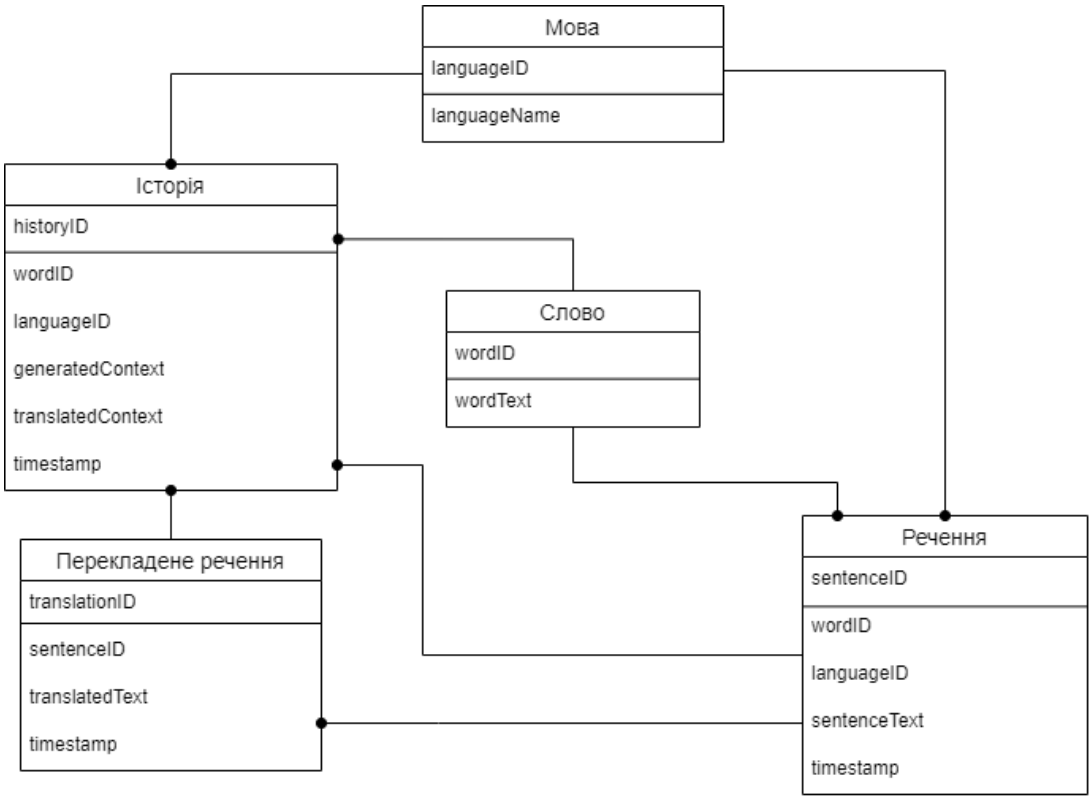
Додаток Б

Схема алгоритму роботи модуля



Додаток В

UML-діагарма класів



Додаток Г

Апробація отриманих результатів

МОЛОДІЖНА НАУКОВА ЛІГА

Громадська організація «Молодіжна наукова ліга».
Номер запису в Реєстрі громадських об'єднань: 1506433.
Адреса: вул. Зодчих, буд. 40, офіс 103; м. Вінниця, Вінницька обл., 21037
Організація функціонує як відокремлений підрозділ ТОВ «UKRLOGOS Group».
ЄДРПОУ: 44574526
IBAN: UA783052990000026003016111950
Банк ВФ АТ КБ «ПриватБанк»; МФО 305299
Свідоцтво суб'єкта видавничої справи: ДК № 7172 від 21.10.2020.

Д О В І Д К А

ПРО ПРИЙНЯТТЯ ТЕЗ ДО ПУБЛІКАЦІЇ

17.06.2024

Шановний(і) авторе(и):
Дідух Данило Романович,

Організаційний комітет з радістю повідомляє, що тези «ПРОГРАМНИЙ МОДУЛЬ ДЛЯ ВИВЧЕННЯ ІНОЗЕМНОЇ МОВИ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ» прийняті до публікації в збірнику за матеріалами VI Всеукраїнської студентської наукової конференції «Експериментальні та теоретичні дослідження в контексті сучасної науки» (21.06.2024, м. Рівне, Україна).

Опубліковані тези будуть доступні з 21.06.2024 за посиланням:
https://archive.liga.science/index.php/conference-proceedings/issue/view/ukr-21.06.2024

. . . . . .

Електронні сертифікати учасників конференції та подяки науковим керівникам будуть доступні з 21 червня. Розсилка замовлених друкованих примірників, сертифікатів та подяк відбудеться до 5 липня.

Конференцію зареєстровано Державною науковою установою «УкрІНТЕІ» в базі даних науково-технічних заходів України та інформаційному бюлетені «План проведення наукових, науково-технічних заходів в Україні» (Посвідчення № 34 від 05.01.2024).

З повагою,

Директор Молодіжної наукової ліги
Голова оргкомітету конференції
ІГОР КОРЕНЮК

ПРОГРАМНИЙ МОДУЛЬ ДЛЯ ВИВЧЕННЯ ІНОЗЕМНОЇ МОВИ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ

Дідух Данило Романович
здобувач вищої освіти факультету
комп'ютерних інформаційних технологій
Західноукраїнський національний університет, Україна
Науковий керівник: Дорош Віталій Іванович
викладач кафедри інформаційно-обчислювальних систем і управління
Західноукраїнський національний університет, Україна

У сучасному світі зростаюча глобалізація і міжнародна співпраця вимагають вільного володіння декількома мовами, а традиційні методи навчання часто не задовольняють потреби сучасних користувачів. Також, швидкий розвиток технологій мобільного та електронного навчання створює попит на інтерактивні, доступні та ефективні програми. Розробка програмного модуля для вивчення іноземної мови на основі штучного інтелекту відповідає сучасним викликам і потребам в галузі освіти, сприяючи не тільки підвищенню якості освіти, але й збільшенню доступу до неї для широкого кола людей.

Створення програмного модуля на мові Swift, який дозволить користувачам ефективно вивчати іноземну лексику через контекстуалізоване використання слів у реченнях. Модуль буде використовувати можливості мікро-сервісів, таких як GPT 3.5 та DeepL, для генерації прикладів речень з різними контекстами, що дозволить користувачам глибше зрозуміти вживання слова у речені.

Для кращого розуміння порядку взаємодій між користувачем, інтерфейсом користувача (UI), модулем обробки запитів (Request Processor), сервісом GPT-3.5, сервісом DeepL та базою даних SwiftData яка згадується у роботі [1] представлені часові системні діаграми, які зображенні на (рис. 1)

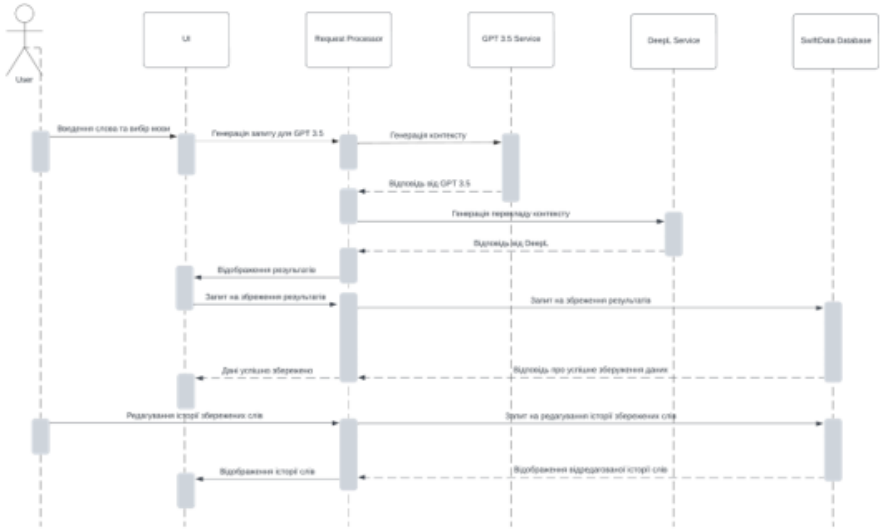


Рис. 1– **Часові системні діаграми**

Головна сторінка додатку (рис. 2) дозволяє користувачам вводити слово для отримання прикладів речень та їх перекладів.

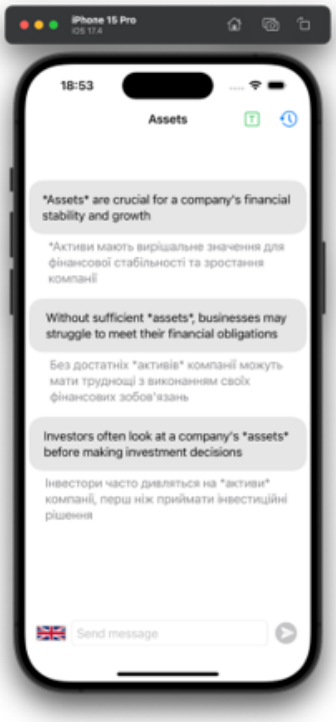


Рис. 2 – **Інтерфейс головної сторінки програмного модуля**

На сторінці «History» програмного модуля (рис. 3) користувачу надається доступ до журналу раніше виконаних запитів щодо генерації речень і їх перекладів.

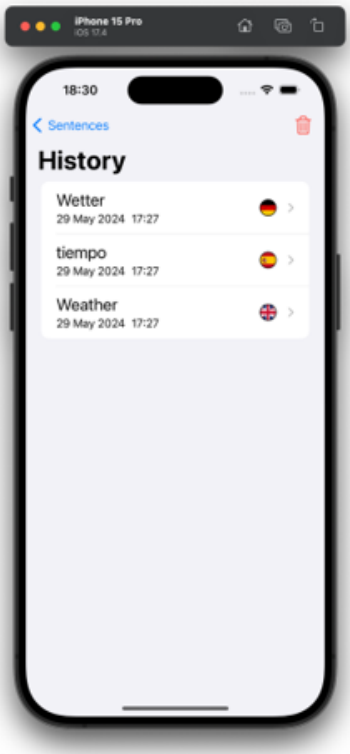


Рис. 3 – **Інтерфейс сторінки історії запитів програмного модуля**

Результатом проведених досліджень і розробок став програмний модуль, здатний значно полегшити процес вивчення іноземних мов, зокрема засвоєння нової лексики через контекстуалізоване використання слів. Завдяки інтеграції сучасних мікро-сервісів і створенню зручного інтерфейсу, ця програма може бути успішно використана як індивідуальними учнями, так і в освітніх установах для навчання мовам. Ця робота є вагомим внеском у галузь розробки програмних засобів для навчання і надає нові можливості для покращення методик вивчення іноземних мов.

Список використаних джерел

1. Apple (2020). Swift Programming Language Guide. Apple Developer

2. Apple (2020). SwiftUI Essentials - Creating and Combining Views. Apple Developer

3. Ray Wenderlich (2021). Learn SwiftUI. Ray Wenderlich

4. Hacking with Swift (2021). 100 Days of SwiftUI. Hacking with Swift