МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В.Н.Каразіна

Факультет математики і інформатики

Кафедра теоретичної та прикладної інформатики

Кваліфікаційна робота

магістр

на тему «Розробка мобільного додатку для адаптивного вивчення англійської мови з використанням методів штучного інтелекту»

Виконав: студент 2 курсу, групи МФ-61

спеціальність 122 «Комп’ютерні науки»

освітньо-наукова програма «Інформатика»

Радченко К. С.

Керівник

Дейнега О. А.

(прізвище та ініціали)

Рецензент_____

(прізвище та ініціали)

Харків – 2026 року

АНОТАЦІЯ

У даній магістерській роботі досліджується задача створення інтелектуальної освітньої екосистеми для персоналізованого вивчення іноземних мов. Запропоновано підхід, заснований на інтеграції мобільного застосунку з мовними моделями з використанням OpenAI API, що дозволяє реалізувати динамічну генерацію навчального контенту в реальному часі. Особливу увагу приділено методам промпт-інжинірингу та використанню структурованих виводів (JSON Schema) для забезпечення технічної надійності та методичної коректності генерованих завдань.

Експериментальне дослідження проведено з метою оптимізації балансу між якістю генерації та швидкістю роботи системи. У роботі проаналізовано ефективність використання різних моделей (зокрема серії GPT-4o) та параметрів генерації, таких як температура та Top-P. Якість результатів оцінювалась за сукупністю метрик, серед яких середня затримка відповіді (Average Time per Request) для оцінки мобільного UX, а також семантична близькість (Mean Topic Similarity), розрахована через косинусну відстань між ембедінгами запиту та результату за допомогою моделей text-embedding-3-small/large.

Отримані результати показують, що використання низьких значень температури (0.3) у поєднанні з JSON-схемами забезпечує стабільну генерацію без «галюцинацій» та помилок парсингу. Аналіз латентності підтвердив, що обрана архітектура дозволяє утримувати час відповіді в межах прийнятного психологічного порогу для мобільних користувачів (близько 10-12 секунд), зберігаючи при цьому високу релевантність навчальних матеріалів обраній темі та рівню складності за шкалою CEFR.

Таким чином, розроблений застосунок є ефективним інструментом для адаптивного навчання, який мінімізує ручну працю методистів і

забезпечує користувачам індивідуальну освітню траєкторію. Запропоновані рішення можуть бути використані у задачах розробки сучасних EdTech-проєктів та систем автоматизації створення контенту.

Загальна характеристика роботи: робота містить вступ, 1 частину, висновки та список використаної літератури. Кількість сторінок - 43, кількість ілюстрацій - 5, кількість використаних джерел - 14.

Ключові слова: штучний інтелект, мобільний додаток, адаптивне навчання, OpenAI API, промпт-інжиніринг, ембедінги, JSON Schema, косинусна подібність, англійська мова.

ABSTRACT

This master's thesis explores the task of creating an intelligent educational ecosystem for personalized foreign language learning. An approach based on integrating a mobile application with large language models via the OpenAI API is proposed, enabling the dynamic generation of educational content in real-time. Particular attention is paid to prompt engineering methods and the use of structured outputs (JSON Schema) to ensure technical reliability and the methodical correctness of the generated exercises.

An experimental study was conducted to optimize the balance between generation quality and system performance speed. The work analyzes the effectiveness of using various models (specifically the GPT-4o series) and generation parameters, such as temperature and Top-P. The quality of the results was evaluated using a set of metrics, including Average Time per Request to assess mobile UX, as well as Mean Topic Similarity, calculated via the cosine distance between the embeddings of the query and the result using text-embedding-3-small/large models.

The obtained results show that the use of low temperature values (0.3) in combination with strict JSON schemas provides stable generation without "hallucinations" or parsing errors. Latency analysis confirmed that the chosen architecture allows response times to be maintained within the acceptable psychological threshold for mobile users (approximately 10-12 seconds), while maintaining high relevance of the educational materials to the selected topic and CEFR difficulty level.

Thus, the developed application is an effective tool for adaptive learning that minimizes the manual labor of methodologists and provides users with an

individual educational trajectory. The proposed solutions can be utilized in the development of modern EdTech projects and content automation systems.

General characteristics of the work: the work contains an introduction, 1 part, conclusions, and a list of used literature. Number of pages – 43, number of illustrations – 5, number of sources used – 14.

Keywords: artificial intelligence, mobile application, adaptive learning, OpenAI API, prompt engineering, embeddings, JSON Schema, cosine similarity, English language

Зміст

АНОТАЦІЯ..... 2
ABSTRACT..... 4
Зміст..... 6
1. ВСТУП..... 7
1.1 Формулювання мети роботи, задач та обґрунтування актуальності теми..... 7
1.2 Стислий огляд відомих результатів..... 8
1.3 Відомості про одержані результати та їх новизна..... 8
2 ОСНОВНА ЧАСТИНА..... 10
2.1 Постановка задачі та вимоги до системи..... 10
2.2 Огляд підходів для предиктивного підбору слів..... 13
2.2.1 Метод Еббінгауза..... 14
2.2.2 Огляд альтернатив метода Еббінгауза..... 15
2.3 Архітектура проекту..... 16
2.3.1 Основи функціонування великих мовних моделей (LLM)..... 17
2.3.2 Структурна схема взаємодії компонентів..... 26
2.3.3 Алгоритм навчання на стороні клієнта..... 28
2.3.4 Переваги обраної архітектури..... 29
2.4 Аналіз ефективності моделі..... 30
2.4.1 Опис обраних моделей..... 30
2.4.2 Опис альтернативних моделей..... 30
2.4.3 Аналіз обраних моделей..... 35
2.5 Результати тестування..... 38
2.5.1 Тестування обраних нейромереж..... 38
2.5.2 Тестування серверної частини (Back End)..... 39
2.5.3 Тестування мобільного додатку..... 39
2.5.4 Комплексне інтеграційне тестування..... 40
ВИСНОВКИ..... 41
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 44

1. ВСТУП

1.1 Формулювання мети роботи, задач та обґрунтування актуальності теми

Актуальність теми. У сучасному глобалізованому світі володіння англійською мовою є критичною вимогою для професійної реалізації та доступу до світових інформаційних ресурсів. Проте традиційні методи навчання часто ігнорують індивідуальні особливості студентів: темп засвоєння матеріалу, початковий рівень знань та когнітивні вподобання. Використання штучного інтелекту (ШІ) дозволяє трансформувати статичний освітній контент у динамічну екосистему, яка підлаштовується під користувача в режимі реального часу. Розробка мобільного додатку, що базується на адаптивних алгоритмах, дозволяє зробити процес навчання максимально ефективним, виключаючи проблему втрати мотивації через занадто складні або занадто прості завдання.

Метою роботи є проектування та програмна реалізація мобільного додатку для вивчення англійської мови, який використовує методи адаптивного штучного інтелекту для створення персоналізованого освітнього напряму для навчання.

Для досягнення мети було визначено такі **завдання**:

1. Проаналізувати існуючі додатки для вивчення англійської, їх методи для адаптивного навчання.
2. Розробити архітектуру мобільного додатку, веб сервісу для генерації навчального матеріалу та структури бази даних.
3. Розробити алгоритм адаптації навчального матеріалу, до змінного рівня знань кожного учня індивідуально.

4. Реалізувати програмний продукт та провести тестування його функціональності.

1.2 Стислий огляд відомих результатів

На сьогодні ринок мобільних застосунків для вивчення мов (Duolingo, DUO cards, Vocab) активно впроваджує елементи ШІ. Зокрема, широко використовуються алгоритми інтервальних повторень (Spaced Repetition).

Проте більшість існуючих рішень мають певні обмеження:

- Лінійність: багато додатків пропонують жорстко визначені курси, де адаптивність обмежується лише повторенням пройденого матеріалу.
- Відсутність глибокої контекстуалізації: чат-боти в таких додатках часто працюють за заздалегідь прописаними сценаріями, що не дозволяє вільно практикувати живу мову.
- Поверхневий аналіз помилок: існуючі системи часто констатують факт помилки, але не аналізують її природу для корекції подальшого плану навчання.
- Доступність матеріалу: у більшості подібних додатків безкоштовний доступ до матеріалів або продвинутих методів навчання є обмеженим, через що за повний доступ до аплікацій потрібно платити.

1.3 Відомості про одержані результати та їх новизна

В ході виконання роботи було отримано такі **результати**:

1. Розроблено адаптивну модель навчання, яка на відміну від класичних систем враховує не лише правильність відповідей, а й рівень користувача, коефіцієнт запам’ятовування кожного слова, аналогічними граматичними конструкціями.

2. Інтегровано методи машинного навчання, а саме Крива Ебінгаузера, вагова вірогідність для динамічного підбору матеріалу за рівнем знань студента.

3. Реалізовано систему налаштувань для більш точних стартових значень, базованих на суб’єктивній оцінці користувача.

Наукова та практична новизна полягає у вдосконаленні алгоритму підбору навчального контенту, що дозволяє скоротити час на засвоєння нової лексики, шляхом предиктивного моделювання “зони найближчого розвитку” користувача. Створений додаток є готовим інструментом, який може бути використаний як для самостійного навчання, так і як допоміжний засіб у межах змішаного навчання (blended learning). Генерація словнику на вузькі теми / пертин тем / актуально для IT.

2 ОСНОВНА ЧАСТИНА

2.1 Постановка задачі та вимоги до системи

Постановка задачі. Самонавчання вимагає переходу від уніфікованих навчальних планів до персоналізованих траєкторій, що базуються на індивідуальних можливостях учня. У контексті вивчення іноземних мов ключовою проблемою є ефективне запам'ятовування лексики. Традиційні додатки часто пропонують надлишкове повторення відомого матеріалу або занадто складні завдання, що призводить до втрати мотивації.

Метою даної роботи є розробка мобільної інтелектуальної системи, яка функціонує як адаптивний репетитор. Система повинна спостерігати за взаємодією користувача з контентом, аналізувати успішність засвоєння матеріалу та динамічно коригувати навчальний план. В основі підходу лежить використання методів штучного інтелекту для моделювання "кривої забування" та предиктивного аналізу знань.

Особливістю даної роботи є реалізація алгоритму адаптивності, який враховує не тільки бінарний результат ("правильно/неправильно"), але й складність слова, контекст помилок та час забування матеріалу. Це дозволяє системі працювати в умовах обмеженого часу користувача, максимізуючи обсяг засвоєної інформації за одну сесію.

Процес адаптивного навчання в системі має декілька ключових аспектів:

- **Спостереження (Observations)** – додаток збирає дані про кожну відповідь користувача: правильність, рівень матеріалу, тема матеріалу.

- **Динамічне керування складністю** – на основі зібраних даних система змінює ваги об'єктів у базі знань. Слова з низьким рівнем засвоєння з'являються частіше, тоді як добре засвоєний матеріал повторюється значно рідше.
- **Генерація контенту** – використання ШІ-моделей (зокрема інтеграція з API сучасних мовних моделей) для створення слів, що відповідають поточному лексичному запасу користувача.
- **Навчальний епізод** – кожна сесія навчання є завершеним циклом, який починається з актуалізації раніше вивченого і закінчується введенням обмеженої порції нового матеріалу.

Завдання передбачає:

1. Реалізацію мобільного клієнта з інтуїтивним інтерфейсом для взаємодії з навчальним контентом.
2. Розробку інтелектуального модуля оцінки знань на основі алгоритмів інтервальних повторень (Spaced Repetition).
3. Створення гнучкої бази даних для зберігання прогресу та метаданих кожного навчального об'єкта.
4. Впровадження механізму генерації контекстуальних завдань за допомогою методів ШІ.
5. Формування системи метрик для оцінки ефективності навчання (коефіцієнт утримання знань, швидкість прогресу).

Обмеження системи

Для забезпечення коректного навчання та стабільної роботи ШІ-модуля накладаються наступні обмеження:

1) Обмеження контенту:

   - **Обсяг бази знань:** початковий словниковий запас обмежений рівнями A1–C2 згідно з класифікацією CEFR для забезпечення точності тестування.

   - **Тематична цілісність:** навчальні матеріали групуються за семантичними полями (темами), щоб уникнути хаотичного накопичення не пов'язаних слів.

   - **Обмеження обсягу матеріалу:** кількість генерації слів за одну ітерацію обмежується діапазоном від 5 до 20 (за замовчуванням - 10) словами для уникнення перевантаження ШІ при генерації навчального матеріалу.

2) Обмеження алгоритму адаптивності:

   - **Дискретність рівнів:** система оперує дискретними рівнями складності, щоб уникнути занадто різких стрибків у навантаженні на користувача.

   - **Часові ліміти:** алгоритм інтервальних повторень обмежений мінімальним кроком (наприклад, не частіше ніж раз на годину) для запобігання ефекту "перенавченості".

Вимоги до системи

Функціональні вимоги:

- Додаток має зберігати навчальний матеріал у локальній базі даних девайсу
- Алгоритм повинен автоматично перераховувати дату наступного повторення слова після кожної відповіді.
- Навчальна сесія має бути зацикленную і з кожною генерацією доповнювати базу даних.

Нефункціональні вимоги:

- **Персоналізація:** алгоритм повинен бути унікальним для кожного профілю користувача.
- **Продуктивність:** час обробки відповіді та оновлення черги завдань не повинен перевищувати 200 мс.
- **Доступність (Offline mode):** основний функціонал повторення вже завантажених слів має працювати без активного підключення до мережі.
- **Масштабованість:** архітектура повинна дозволяти легке додавання нових мовних пар або нових типів ШІ-моделей (наприклад, перехід від статистичних методів до нейронних мереж).

2.2 Огляд підходів для предиктивного підбору слів

Процес предиктивного підбору навчального контенту базується на спробі моделювання когнітивних процесів людської пам'яті. Основна задача інтелектуального модуля додатка – передбачити момент, коли

ймовірність згадування слова користувачем опускається нижче критичного порогу, і саме в цей момент запропонувати матеріал для повторення.

Фундаментом для побудови адаптивних систем навчання є крива забування Еббінгауза. Вона описує експоненціальну втрату затриманої в пам'яті інформації з плином часу. В даній роботі було удосконалено метод за допомогою використання Spaced repetition, де кожне успішне повторення збільшує показник запам’ятовування, роблячи нахил кривої забування більш пологим. У сучасних ШІ-системах цей підхід трансформується з жорстких графіків у динамічні моделі, де інтервали обчислюються індивідуально для кожного слова та Weighted probability, де у якості вагів було використано різницю між правильними та неправильними спробами.

2.2.1 Метод Еббінгауза

Фундаментом для побудови адаптивних систем навчання є крива забування Германа Еббінгауза. Експериментально доведено, що процес забування нової інформації має експоненціальний характер: найбільша втрата даних відбувається у перші години після навчання, після чого швидкість забування поступово сповільнюється.

Математично цей процес описується формулою:

$$R = e^{-1\frac{t}{S}}$$

де:

- R, ретенція (частка збереженої інформації, $0 \leq R \leq 1$)
- t, час, що минув з моменту останнього вивчення

- S, відносна міцність пам’яті, яка залежить від якості первинного навчання та кількості повторень.

Згідно з цією моделлю, кожне наступне успішне повторення збільшує показник S, що робить нахил кривої забування більш пологим. Це означає, що інформація утримується в пам’яті довший час.

2.2.2 Огляд альтернатив метода Еббінгауза

Під час вибору математичної моделі для побудови алгоритму адаптивного навчання було розглянуто кілька альтернативних підходів. Метою було знайти баланс між точністю моделювання когнітивних процесів та обчислювальною складністю, що є критичним для мобільного застосунку.

1. **Метод статичних інтервалів**. Спрощена модель, де інтервали повторення є фіксованими (наприклад, 1, 3, 7, 30 днів) незалежно від успішності користувача. Цей метод не був обраний через ігнорування індивідуальних особливостей запам’ятовування.
2. **Байєсівське відстеження знань** (Bayesian Knowledge Tracing - BKT). BKT розглядає навчання як прихований марковський процес. Модель намагається передбачити ймовірність того, що користувач "опанував" концепт, на основі історії його відповідей. Не було обрано через високу алгоритмічну складність та необхідність великого набору даних для калібрування параметрів.
3. **Евристичні алгоритми.** Використання випадкових або циклічних черг (наприклад, перемішування словника без аналізу часу відповіді). Не було обрано через відсутність наукового підґрунтя і не гарантують довготривалого запам’ятовування. Метод є більш шкідливим ніж корисним для кінцевого користувача.

2.3 Архітектура проекту

Архітектура системи базується на клієнт-серверній моделі, де мобільний додаток виконує роль інтерфейсу збору даних та візуалізації, а серверний модуль відповідає за складні обчислення, управління станом моделі та інтеграцію з мовними моделями (LLM).

Нижче наведене схематичне зображення архітектури проекту:

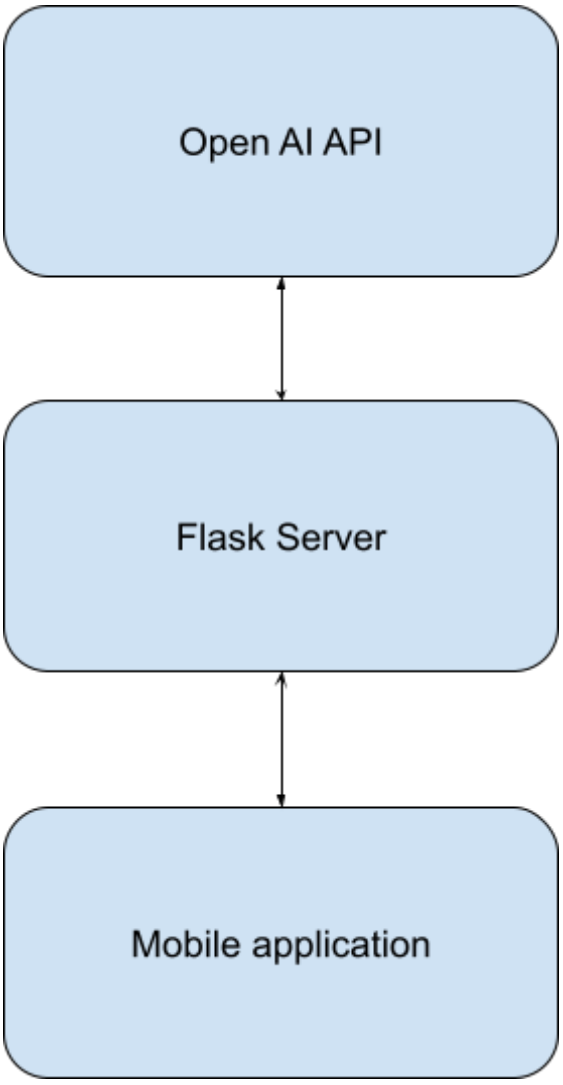

Рис. 2.1. Архітектура проєкту

2.3.1 Основи функціонування великих мовних моделей (LLM)

Великі мовні моделі (LLM – Large Language Models) представляють собою підклас моделей глибокого навчання, навчених на масивах текстових даних обсягом у терабайти. Основною метою таких моделей є передбачення наступного елемента послідовності (токена) на основі вхідного контексту. У контексті даної дипломної роботи, LLM використовуються не просто як чат-боти, а як генеративні рушії для створення персоналізованого навчального контенту.

Сучасні LLM базуються на архітектурі Transformer (представленій Google у 2017 році), ключовою особливістю якої є механізм Self-Attention (механізм “самоуваги”). Цей механізм дозволяє моделі динамічно визначати вагу (значущість) кожного слова у реченні відносно всіх інших слів, незалежно від їхньої відстані у тексті.

Завдяки цьому, модель ефективно вловлює:

Семантичні зв'язки: розуміння синонімів та контекстуальних відтінків слів.

Синтаксичні залежності: правильна побудова граматичних конструкцій.

Довготривалі залежності: здатність зберігати логіку розповіді або діалогу на великих проміжках тексту.

У розробленій системі LLM виконує роль “генератора контексту”. На відміну від статичних баз даних, де приклади вживання слів є фіксованими, LLM дозволяє реалізувати контекстуальну адаптацію.

Основні переваги використання LLM у нашому проєкті:

Адаптивність до рівня знань: За допомогою інструкцій (промптів) ми можемо обмежувати словниковий запас моделі, змушуючи її генерувати речення для рівня A1, B1 або B2 та завчасно обраної теми, гарантуючи, що учень розуміє допоміжну лексику в прикладі.

Мультимодальність запитів: LLM дозволяє не лише генерувати слова, але й створювати супровідні пояснення, переклади та тестові питання, що значно зменшує час на підготовку контенту методистами.

Сімейство моделей OpenAI, об’єднаних під загальною назвою GPT (Generative Pre-trained Transformer), являє собою клас великомасштабних мовних моделей, що використовують методи глибокого навчання для обробки та генерації текстового контенту. В основі роботи цих моделей лежить архітектура Transformer, яка забезпечила значний прогрес у завданнях обробки природної мови завдяки механізму уваги.

Моделі GPT базуються на декодерній частині архітектури Transformer. Основний принцип їх функціонування полягає у ймовірнісному прогнозуванні наступного елемента послідовності (токена) на основі вхідного контексту.

Процес створення цих моделей складається з двох ключових етапів:

1) **Попереднє навчання (Pre-training):** Модель тренується на колосальних масивах текстових даних, вивчаючи структуру мови, логіку побудови речень та фактичні залежності. На цьому етапі модель формує статистичні представлення про зв’язки між словами.
2) **Fine-tuning та RLHF (Reinforcement Learning from Human Feedback):** Після попереднього навчання моделі проходять процедуру доналаштування за участі людини. Метод RLHF дозволяє “навчити” модель слідувати інструкціям та бути максимально

корисною для користувача. Це перетворює модель із простого “прогнозиста тексту” на “інтелектуального асистента”.

OpenAI надає доступ до лінійки моделей, які розрізняються за кількістю параметрів, архітектурними оптимізаціями та обчислювальною вартістю.

Одним із критичних аспектів використання моделей OpenAI є робота з контекстом. Оскільки кожна модель має обмеження на розмір вхідних даних, серверна частина у проекті бере на себе завдання формування оптимального промпту. На сервері мінімізується надмірність тексту в запиті, передаючи лише ті дані, які необхідні для виконання конкретного лінгвістичного завдання, що дозволяє економити обчислювальні ресурси та час генерації.

Embeddings – це спосіб перетворення об'єктів складної природи (як-от слова, фрази або цілі документи) у вектори чисел з плаваючою комою у багатовимірному просторі.

Комп'ютери не розуміють значення слів – вони працюють лише з числами. Раніше (у часи One-Hot Encoding або TF-IDF) слова були представлені як розріджені вектори, де більшість значень – нулі. Це було неефективно, бо такі методи не враховували контекст або синонімію.

Ембедінги – це щільні вектори (dense vectors). У моделях серії text-embedding-3 кожен текст перетворюється на масив чисел (наприклад, 1536 розмірностей для small або 3072 для large).

text-embedding-3-small: Оптимізована модель для завдань, де критичними є швидкість та економічна ефективність. Вона має стандартну розмірність 1536 вимірів.

text-embedding-3-large: Найпотужніша модель, розроблена для глибокого семантичного розуміння. Вона підтримує розмірність до 3072 вимірів, що дозволяє їй вловлювати складні нюанси значень, контекстуальні зв'язки та професійну термінологію значно краще за попередні версії.

Чому були обрані саме ці моделі:

- **Єдиний екосистемний підхід.** Оскільки основний контент (навчальні завдання) генерується за допомогою моделей GPT-4o, використання рідних ембедінгів від OpenAI забезпечує максимальну семантичну сумісність. Простір знань, у якому працює модель-генератор, збігається з простором, у якому модель-оцінювач проводить вимірювання (Mean Topic Similarity).
- **Гнучкість.** Було використано small embedding для динамічного індексування контенту в базі даних застосунку, де швидкість відповіді є пріоритетом. Було використано large embedding як "золотий стандарт" для верифікації якості згенерованого тексту (Mean Topic Similarity). Вища розмірність large дозволяє мінімізувати помилки при оцінці того, чи відповідає відповідь ШІ заданій навчальній темі.
- **Технологічна перевага.** Завдяки цій технології можна динамічно зменшувати розмірність ембедінгів без втрати їхньої семантичної цілісності.
- **Висока стійкість до багатомовності.** Обидві моделі демонструють покращену роботу з мовами, відмінними від англійської, що є критично важливим для освітнього проєкту, оскільки навчальний контент може включати змішування мов або специфічну лінгвістичну термінологію.

Концепція промпт-інжинірингу та параметри генерації

У сучасних системах, що базуються на великих мовних моделях (LLM), **промпт (prompt)** є основним інструментом керування поведінкою штучного інтелекту. Це текстовий вхідний сигнал (інструкція або запит), який задає контекст для моделі та визначає формат, стилістику та зміст вихідних даних. Процес розробки та оптимізації таких запитів називається **промпт-інжинірингом**.

Основні принципи побудови ефективного промпту

Для забезпечення стабільної якості генерації навчального контенту в проєкті було використано наступні методології:

- **Визначення персони (Persona Prompting):** Моделі призначається конкретна роль. Це дозволяє моделі адаптувати лексику, тон подання інформації та точність формулювань під стандарти для заданої ролі.
- **One-shot Prompting:** Надання моделі одного конкретного прикладу «запит-відповідь». Це допомагає ШІ зрозуміти структуру бажаного результату (наприклад, формат JSON), що критично важливо для подальшого парсингу даних клієнтом.
- **Few-shot Prompting:** Надання кількох прикладів (зазвичай від 3 до 5). Цей підхід є найбільш надійним для складних завдань, оскільки він дозволяє моделі вловити патерни, логіку побудови та нюанси складності мови.

Технічні параметри генерації

Крім текстової інструкції, на результат генерації суттєво впливають конфігураційні параметри API, які дозволяють керувати ступенем творчості та передбачуваності моделі:

1) **Temperature (Температура):** Параметр, що контролює рівень випадковості генерації.
    a) Значення близько **0** роблять модель детермінованою (вона завжди обирає найбільш імовірне наступне слово), що корисно для сухих технічних перекладів
    b) Значення **0.7–1.0** додають моделі «креативності». У проєкті було обрано значення, близьке до 0, щоб забезпечити різноманітність навчальних прикладів, але уникнути фактичних помилок.
2) **Top-P (p-value або Nucleus Sampling):** Альтернатива температурі, яка обмежує вибір моделі лише найбільш імовірними токенами, сумарна ймовірність яких складає *p*.
    a) Наприклад, при **Top-P = 0.9** модель розглядає лише ті слова, які разом складають 90% ймовірності. Це дозволяє відсікати малоймовірні слова, підвищуючи логічну зв'язність тексту.

Нижче буде наведено скріншот промпту, який був застосований у проєкті:

```
system = (
    "You generate English vocabulary lists.\n"
    "Translations MUST be Ukrainian.\n"
    "Examples MUST be natural English and appropriate for the requested CEFR level.\n"
    "No profanity. No extra keys. No markdown."
)

user = f"""
Create {n_words} English vocabulary items for CEFR level {level} about this topic:

TOPIC: {topic}

Rules:
- Exactly {n_words} items.
- Prefer single words or short terms (1–3 words max).
- Difficulty must match {level}.
- Translation must be Ukrainian.
- Provide 2–3 English example sentences per item.
- give 4 answers using translation as a right answer for each item. (1 right answer and 3 wrong answers, right answer must be at random position)
Return the result as JSON with key "items".
"""
```

Рис 2.2. Основний промпт

Для забезпечення високої якості генерації навчального контенту в проєкті було розроблено багаторівневу систему взаємодії з великою мовною моделлю (LLM), яка базується на принципах програмованого промпт-інжинірингу та суворого дотримання JSON-схем.

Принципи побудови та структура промпту

У системі використано підхід розділення інструкцій на **System Prompt** (системна роль) та **User Prompt** (завдання користувача):

- **System Persona (Рольова модель):** Моделі чітко призначено роль генератора англійського словника ("You generate English vocabulary lists"). Це обмежує простір варіантів відповідей і фокусує модель на конкретній лінгвістичній задачі. Системна інструкція також містить жорсткі обмеження (No profanity, No extra keys, No markdown), що гарантує безпеку та чистоту вихідних даних.

- **Контекстуалізація та параметризація:** Користувацький запит (User Prompt) динамічно формується за допомогою змінних (topic, level, n_words). Це дозволяє системі бути гнучкою та адаптуватися під індивідуальну навчальну програму користувача, зберігаючи при цьому сталу структуру.

- **Zero-shot з суворою специфікацією:** Замість надання прикладів (Few-shot), у даному промпті використано метод **Strict Instruction**. Оскільки структура результату дуже складна (JSON зі вкладеними масивами відповідей та прикладів), замість текстових прикладів використано механізм **Structured Outputs (json_schema)**. Це гарантує, що модель поверне об'єкт, який на 100% відповідає програмному інтерфейсу (API) мобільного застосунку.

Технічні параметри генерації

Для досягнення максимальної стабільності роботи алгоритму було застосовано специфічні параметри API:

- **Temperature (0.3):** Для генерації словникових списків було обрано низьке значення температури. Це робить модель більш передбачуваною та зосередженою на фактах. Низька температура мінімізує ризик "галюцинацій" у перекладах та забезпечує відповідність прикладів речень заданому рівню складності CEFR.
- **Structured Outputs (Strict JSON Mode):** Це ключовий параметр, який змушує модель слідувати схемі vocab_list. Використання strict: True гарантує, що кожне поле (переклад, приклади, варіанти відповідей) буде присутнє у відповіді, що унеможливлює виникнення помилок при парсингу даних у мобільному застосунку.
- **Reasoning Effort (Medium):** Для моделей, що підтримують логічне виведення (reasoning), встановлено середній рівень зусиль. Це забезпечує глибший аналіз контексту теми та підбір найбільш влучних синонімів для неправильних варіантів відповідей, що підвищує якість навчального тестування.

Поєднання рольового моделювання, параметричного керування температурою та суворої валідації JSON-схем дозволило перетворити імовірнісну модель ШІ на надійний інструмент генерації навчального контенту, готовий до використання у продуктовому середовищі.

Перед тим як обрати даний промпт було розглянуто альтернативний промпт:

```
'''Create 10 English vocabulary items for CEFR level {level} about this topic:

    TOPIC: {topic}

    Rules:
    - Prefer single words or short terms (1–3 words max).
    - Difficulty must match {level}.
    - Translation must be Ukrainian.
    - Provide 2–3 English example sentences per item.
    - give 4 answers using translation as a right answer for each item.
    Return the result as JSON with key "items".
    """
```

Рис 2.3. Альтернативный промпт

Аналіз недоліків альтернативного підходу:

- **Відсутність системної ролі та персони.** У цьому варіанті не задано роль «викладача» чи «генератора словника». Це призводить до того, що модель може використовувати занадто складні або, навпаки, занадто побутові речення, які не відповідають методичним вимогам заданого рівня.
- **Невизначеність структури (Відсутність JSON-схеми)** Промпт просто просить «формат JSON». Без використання **Structured Outputs (json_schema)** модель може:
    - Додати зайвий текст перед або після JSON (наприклад: *"Ось ваш список:"*)
    - Повертати різні назви ключів
    - Помилитись у вкладеності масивів. Це призведе до критичної помилки при спробі розпарсити дані в Android-застосунку
- **Відсутність жорстких обмежень контенту.** У промпті не вказано заборону на нецензурну лексику (No profanity) або обмеження довжини термінів (1–3 words max). У результаті модель може

згенерувати цілі фразеологізми або речення замість окремих слів, що не підходить для інтерфейсу мобільного тренажера.

- **Проблема з наданням правильних відповідей.** Через те що не вказано надавати правильну відповідь у випадковому місці, модель завжди повертала спочатку правильну відповідь, потім 3 неправильних.

2.3.2 Структурна схема взаємодії компонентів

Система складається з трьох ключових рівнів:

1) **Mobile Client (Android, Kotlin):** Реалізує логіку відображення контенту та локального збереження проміжних станів. Додаток фіксує кожну дію користувача (метрики часу, правильність відповіді), зберігає у локальній базі даних телефона для можливості подальшої обробки і відображення даних отриманих від сервера.

2) **Backend Service (Flask, Python):** Виступає постачальником даних (слів) до клієнта. Він містить реалізацію логіку формування запитів до зовнішніх ШІ-сервісів.

3) **Intelligence Layer (OpenAI API):** Використовується для динамічної генерації навчального контенту (contextual sentence generation), що дозволяє створювати слова, адаптовані під поточний рівень користувача.

Нижче наведені скріншоти мобільного додатку з їх описом.

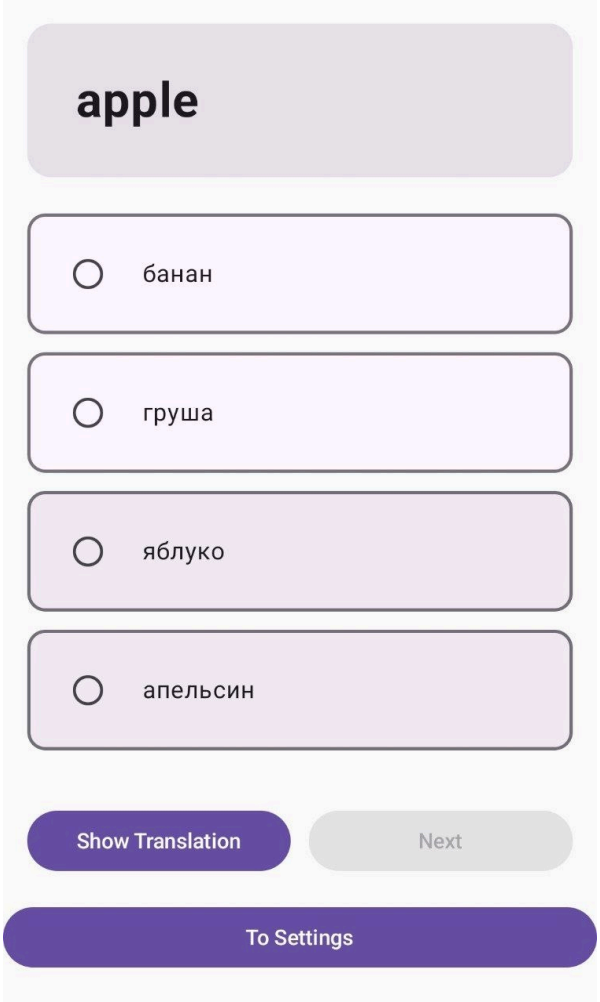

Рис 2.4. Основний екран.

Основний екран складається з двох частин:

1. **Нефункціональний контент:** невеликий блок де розташовується слово, переклад якого треба було визначити

2. **Функціональний контент: блок із варіантами відповідей**, має три основні статуси, за замовчуванням (4 варіанти відповідей без виділень), коли надана правильна відповідь вона підсвічується зеленим і при неправильній відповіді неправильна підсвічується червоним, а правильна - зеленим. **Блок навігації** складається з трьох кнопок. Кнопка переходу до наступного слова (Next) стає активною

після того, як користувач надасть відповідь. Кнопка “**Show translation**” показує переклад даного слова і встановлює спробу відповіді, як не правильно. **“To settings”** кнопка відповідає за навігацію до наступного екрану, в якому у користувача є можливість налаштувати параметри генерації слів.

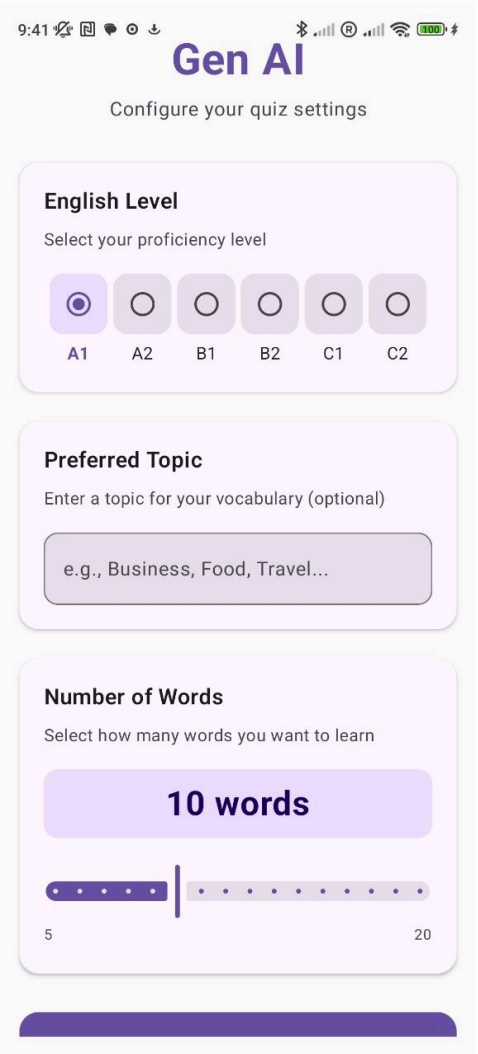

Рис 2.5 Екран налаштувань.

2.3.3 Алгоритм навчання на стороні клієнта

Оскільки математичні обчислення проходять на клієнті, алгоритм адаптивності працює за наступною схемою:

1) **Фіксація події:** Після кожної відповіді вправа передає дані (успіх/помилка, час) у внутрішній контролер для збереження у локальній базі даних.

2) **Локальне обчислення:**

   a) Модуль аналітики на Kotlin оновлює запис у Room DB.

   b) Розраховується нова вага за формулою

   c) Формується список слів у порядку спадання вагів

2.3.4 Переваги обраної архітектури

Було обрано підхід через низку переваг, які я зазначу нижче:

1) **Автономність:** Основна логіка навчання (вибір слів, перевірка знань) працює навіть без доступу до інтернету. Мережа потрібна лише для завантаження нових наборів слів від ШІ.

2) **Приватність:** Детальна історія помилок та прогресу користувача не передається на сервер, що підвищує рівень захисту персональних даних.

3) **Зниження навантаження на сервер:** Flask-сервер не витрачає ресурси на математичні розрахунки для тисяч користувачів, а лише перенаправляє запити до ШІ, що робить систему легко масштабованою.

2.4 Аналіз ефективності моделі

2.4.1 Опис обраних моделей

gpt-4.1-nano: Це оптимізована, легковагова версія моделі, спеціально розроблена для задач з низькою латентністю. Її архітектура передбачає меншу кількість параметрів порівняно з повнорозмірними моделями, що дозволяє досягати мінімального часу відповіді. Дана модель демонструє високу точність у виконанні вузькоспеціалізованих завдань, зокрема при генерації слів та виборі правильних варіантів відповідей.

gpt-5-mini: Модель середнього класу, що позиціонується як компромісне рішення між швидкістю та глибиною генерації. Архітектура gpt-5-mini базується на механізмах стиснення знань, що дозволяє їй зберігати контекстуальну складність при менших витратах ресурсів на обробку одного запиту. Вона була протестована для оцінки того, чи покращує "середня" складність моделі семантичну точність.

gpt-5.4: Модель з високим рівнем параметризації, призначена для виконання складних когнітивних завдань. Вона володіє найбільшими можливостями з точки зору семантичного моделювання, проте вимагає значних обчислювальних ресурсів. У контексті мобільного додатку було важливо перевірити, чи забезпечує її "інтелектуальна перевага" суттєве зростання якості контенту, яке виправдовувало б суттєве зростання часу очікування відповіді користувачем.

2.4.2 Опис альтернативних моделей

Перш ніж обрати **OpenAI API** як основний технологічний стек для реалізації системи, було проведено комплексне дослідження ринку хмарних обчислень та рішень у сфері генеративного штучного інтелекту.

Основна увага приділялася пошуку балансу між швидкістю розробки, масштабованістю інфраструктури та якістю лінгвістичних моделей.

Нижче наведено порівняльний аналіз провідних альтернативних платформ – **Google Cloud Platform (Vertex AI)** та **Amazon Web Services (Amazon Bedrock)**. Порівнювалися такі критичні критерії, як стабільність API, підтримка структурованого виводу даних (JSON Schema), мультимодальні можливості та економічна доцільність використання для мобільних EdTech-застосунків.

Дане порівняння дозволяє обґрунтувати, чому саме архітектура OpenAI на поточному етапі розвитку проєкту забезпечує найбільш ефективну реалізацію алгоритмів адаптивного навчання та семантичної верифікації контенту.

Google Cloud Platform (GCP): Екосистема Vertex AI

Google Cloud фокусується на глибокій інтеграції власних моделей (Gemini) з інфраструктурою TPUs (Tensor Processing Units).

Платформа Vertex AI. Це єдиний робочий простір, який включає Gemini Enterprise Agent Platform. Вона дозволяє не лише викликати API, а й будувати цілі агентні ланцюжки (Agent Studio, Agent-to-Agent Orchestration).

Флагманські моделі:

1) Gemini 3.1 Pro: Основна модель для складних логічних завдань та обробки довгих контекстів (long-running agents).
2) Gemini 3.1 Flash Image (Nano Banana 2): Спеціалізована мультимодальна модель для швидкої генерації візуального контенту.
3) Lyria 3: Професійна модель для генерації аудіо.

4) Anthropic Claude Opus 4.7: Google розширює концепцію "відкритого вибору", інтегруючи найкращі зовнішні моделі безпосередньо у Vertex AI.

Amazon Web Services (AWS): Екосистема Amazon Bedrock

AWS дотримується стратегії “модельної агностичності”. Їхня платформа створена так, щоб ви могли легко перемикатися між різними вендорами (Anthropic, Meta, Mistral, Amazon), не змінюючи код застосунку.

Платформа: Amazon Bedrock

Це повністю керований сервіс для доступу до фундаментних моделей. Головна перевага Bedrock – це Knowledge Bases (керований RAG) та швидка інтеграція з іншими сервісами AWS (Lambda, OpenSearch, Aurora)

Флагманські моделі:

- **Amazon Nova 2 (Lite, Pro, Omni):** Власна лінійка моделей від Amazon, оптимізована під специфічні завдання.
- **Anthropic Claude Opus 4.7 & Claude Mythos:** AWS має дуже тісні партнерські стосунки з Anthropic. Модель Claude Mythos (preview 2026) – це спеціалізована модель для кібербезпеки та аналізу великих баз коду.
- **Llama 4 (Meta):** Відкрита модель, доступна через API Bedrock.

Таблиця 2.1 – Порівняльна характеристика альтернативних моделей

Критерій	Google Cloud (Vertex AI)	AWS (Amazon Bedrock)
Філософія	Вертикальна	Вибір та гнучкість

	інтеграція ("Gemini First")	("Model-as-a-Service")
Власна флагманська модель	Gemini 3.1	ProAmazon Nova 2 Pro/Omni
Спеціалізація	Агентні системи, TPU-прискорення	Enterprise-ready RAG, широкий вибір моделей
Партнерства	Anthropic (Claude)	Anthropic (Claude), Meta (Llama), Mistral
Найкраще для	Команд, що вже в екосистемі Google	Корпорацій, що потребують незалежності від одного вендора

Під час проєктування системи було розглянуто кілька постачальників послуг генеративного ШІ (OpenAI, Google Vertex AI, AWS Bedrock). В результаті аналізу для реалізації MVP (Minimal Viable Product) було обрано OpenAI API. Це рішення ґрунтується на чотирьох ключових інженерних та бізнес-факторах:

Синергія "Генерація + Ембедінги" (Технічна цілісність)

Найвагомішим аргументом стала єдність екосистеми OpenAI. Моделі генерації (GPT-4o/mini) та моделі ембедінгів (text-embedding-3-small/large) розроблялися як єдиний продукт.

Результат: Токенізатор та "ментальна модель" (спосіб інтерпретації мови) у них спільні. Це означає, що оцінка семантичної близькості (Topic Similarity), яка проводиться через ембедінги, є максимально точною для текстів, згенерованих саме цими моделями. Використання моделей від різних вендорів (наприклад, генерація через Anthropic, а ембедінги через

Google) могло б призвести до "семантичного зміщення", де метрики оцінки не відображали б реальну якість контенту.

Оптимальний баланс Latency/Quality (Ефективність для мобільного продукту)

Мобільний застосунок вимагає низького часу відгуку (latency). Моделі серії gpt-4o-mini демонструють найкращі показники швидкості генерації серед усіх доступних на ринку API-рішень при збереженні високої якості виконання складних інструкцій (Instruction Following).

Стандартизація та екосистема (DX – Developer Experience)

OpenAI API є "золотим стандартом" індустрії.

- **Документація:** Якість документації та наявність SDK (Python/Kotlin wrapper) дозволили скоротити час на інтеграцію в декілька разів порівняно з налаштуванням складних VPC-конфігурацій у хмарних провайдерів (GCP/AWS).
- **Спільнота:** У разі виникнення помилок (наприклад, ліміти API, обробка потокових відповідей – Streaming), рішення є легкодоступними в документації або спільноті розробників. Це мінімізувало ризик технічного блокування на етапі розробки.

Гнучкість налаштувань

OpenAI надає найбільш прозорий контроль над параметрами генерації (temperature, top_p, frequency_penalty). Для навчального застосунку, де треба балансувати між креативністю (щоб вправи були цікавими) та детермінізмом (щоб відповіді не були помилковими), ця гнучкість є критичною. Ми маємо можливість тонкого налаштування

«особистості» бота через System Prompt, що значно ефективніше, ніж у багатьох open-source альтернатив.

2.4.3 Аналіз обраних моделей

Для визначення оптимального генератора контенту в рамках розробленої системи було проведено порівняльне тестування трьох версій мовних моделей: gpt-4.1-nano, gpt-5-mini та gpt-5.4. Критеріями оцінки стали:

- **Overall Dictionary Validity (%)** - відсоток коректно сформованих слів, тобто перевірка на існування даного слова у словнику. Це важливий параметр для запобігання вивчення користувачем неіснуючого матеріалу.
- **Overall Level Alignment (%)** - відповідність згенерованого контенту заданому рівню складності (CEFR). За допомогою Zipf (частотою використання слова) ми визначаємо рівень слова і порівнюємо з обраним рівнем складності.
- **Mean Topic Similarity** - семантична близькість згенерованого тексту до навчальної теми (для перевірки використано моделі text-embedding-3-small та text-embedding-3-large). Для кількісного вимірювання цієї метрики розраховується косинусна відстань між вектором початкового запиту та векторами отриманих слів у багатовимірному просторі. Це дозволяє визначити математичну точність відповідності згенерованого контенту заданій темі: чим ближче значення косинусної подібності до одиниці, тим вищою є релевантність отриманих термінів навчальному контексту.

- **Average Time per Request (sec)** - середня затримка відповіді, що є критичним параметром для мобільного користувацького досвіду (UX). Важливість цього параметра зумовлена кількома ключовими чинниками:

    - **Підтримка когнітивного потоку (Flow State):** Навчання вимагає високої концентрації. Затримка відповіді понад 10-12 секунд створює “когнітивний розрив”: поки користувач очікує на згенероване завдання, він втрачає нитку логічних міркувань, відволікається на зовнішні фактори або втрачає інтерес до процесу. Це безпосередньо впливає на показник утримання користувачів (Retention Rate).

    - **Технічна стабільність у мобільних мережах:** Мобільні пристрої часто працюють у нестабільних мережах (перемикання між 4G/5G, слабкий сигнал). Довготривалий запит підвищує статистичну ймовірність виникнення мережевих помилок (Network Timeouts) та переривання сесії, що робить застосунок непридатним для використання в реальних умовах.

    - **Енергоефективність:** Активне очікування відповіді від сервера змушує радіомодуль смартфона перебувати в стані підвищеного споживання енергії. Оптимізація цього часу – це також ефективне використання заряду акумулятора користувача, що є частиною загального комфорту при роботі з застосунком.

Результати тестування представлено у Таблиці 2.2.

Таблиця 2.2 – Порівняльна характеристика моделей генерації контенту

Критерій	gpt-4.1-nano	gpt-5-mini	gpt-5.4
Overall Dictionary Validity	100.00%	98.33%	96.67%
Overall Level Alignment	39.17%	30.83%	40.00%
Mean Topic Similarity(small)	44.89%	38.79%	36.31%
Mean Topic Similarity(large)	43.22%	38.24%	36.07%
Average Time per Request	11.10s	41.62s	21.17s

Аналіз отриманих даних дозволяє зробити наступні висновки:

1. **Латентність (швидкість):** Модель gpt-4.1-nano показала найкращий результат за часом відповіді (11.10 с), що майже в 2 рази швидше за gpt-5.4 та майже в 4 рази швидше за gpt-5-mini. Для мобільного додатку, де користувач очікує швидкого відгуку системи, цей показник є вирішальним.

2. **Якість контенту (Validity & Alignment):** Модель gpt-4.1-nano демонструє ідеальну точність словникового наповнення (100%), що робить її найнадійнішою для навчання. Хоча gpt-5.4 має дещо вищий показник відповідності рівню складності (40.00% проти 39.17%), різниця є незначною і не виправдовує подвоєння часу очікування.

3. **Семантична близькість:** Показники семантичної близькості (Topic Similarity) вказують на те, що gpt-4.1-nano краще структурує відповіді відповідно до навчального контексту. Варто зазначити, що використання моделі ембедінгів text-embedding-3-small показало

стабільно вищі результати семантичної відповідності для всіх тестуваних моделей у порівнянні з large-версією.

З огляду на отримані дані, модель gpt-4.1-nano була обрана як пріоритетна для інтеграції в систему. Вона забезпечує найкращий баланс між якістю навчання (найвища точність та семантична близькість) та швидкістю, що є критично важливим для забезпечення високого рівня задоволеності користувачів мобільного застосунку. Використання важчих моделей, таких як gpt-5-mini або gpt-5.4, призводить до суттєвих затримок у відповіді від системи без відповідного приросту якості навчального матеріалу.

2.5 Результати тестування

Тестування системи проводилось з метою перевірки відповідності функціональним та нефункціональним вимогам, оцінки стабільності взаємодії між компонентами та вимірювання продуктивності. Процес тестування був розділений на чотири етапи: тестування нейромоделей, тестування серверної частини (Backend), тестування мобільного інтерфейсу (UI) та комплексне інтеграційне тестування.

2.5.1 Тестування обраних нейромереж

Тестування проходило шляхом порівняння версій моделей (gpt-4.1-nano, gpt-5-mini та gpt-5.4). Основними критеріями були швидкість відповіді та якість генерації. Якість складалась з трьох показників: відповідність словнику, і відповідність темі і рівню. За результатами стало зрозуміло, що оптимальною моделлю буде gpt-4.1-nano, через її оптимальну швидкість обробки запиту та точність наданих даних.

2.5.2 Тестування серверної частини (Back End)

Серверний модуль перевірявся на коректність обробки запитів, стабільність роботи під навантаженням та безпеку передачі даних.

- **API endpoints,** проведено валідацію всіх ендпоінтів. Обробка запитів з некоректними вхідними даними повертали відповідні відповіді.
- **Stress test,** імітація великої кількості одночасних запитів до Flask сервера показала, що сервер здатний успішно обробляти всі запити.
- **Response serialization,** дані повертаються в повному обсязі, корректно.

2.5.3 Тестування мобільного додатку

Тестування мобільного додатку, розробленого за допомогою технології Jetpack Compose, було спрямовано на перевірку чутливості інтерфейсу корректності відображення даних із локальної бази даних Room DB.

- **Responsive UI,** інтерфейс успішно адаптується під різні розміри мобільних телефонів, затримки при зміні станів або взаємодії з окремими елементами відсутні.
- **Database State Management,** проведено тести, що підтверджують негайне оновлення UI при зміні даних у Room DB. Після обчислення алгоритмом SRS нових параметрів слова, інтерфейс миттєво відображає оновлений статус навчання.

2.5.4 Комплексне інтеграційне тестування

Це етап, де перевірялася робота всіх трьох модулів (Android-клієнт, Flask-сервер, OpenAI API) як єдиного ланцюга.

Сценарій тестування: Користувач відповідає на завдання → Клієнт оновлює стан у Room DB → Алгоритм SRS виявляє необхідність генерації нового контенту → Клієнт надсилає запит на Flask → Flask звертається до OpenAI → Отриманий контент зберігається в локальну БД → UI оновлюється.

Результати інтеграційного тестування:

1) **Зв'язність,** обмін даними через Retrofit2 відбувається стабільно. Випадки розриву з’єднання обробляються механізмами повторних спроб.

2) **Цілісність даних,** жоден запит, отриманий від ШІ, не був втрачений при збереженні в Room, структура сутностей відповідає вимогам бізнес-логіки.

3) **Латентність системи,** загальний час від відповіді з сервера до оновлення інтерфейсу складає в середньому 12-14 секунд (з урахуванням часу на обчислення на клієнті та відповіді API), що визнано задовільним для освітнього застосунку.

Загалом, тестування довело, що система є стабільною, масштабованою та здатною забезпечувати безперервний процес навчання користувача, ефективно поєднуючи локальні математичні обчислення з потужностями хмарних нейронних мереж.

ВИСНОВКИ

У ході виконання роботи було реалізовано архітектуру та впроваджено функціональний прототип інтелектуальної системи адаптивного навчання іноземної мови, яка базується на математичних моделях інтервальних повторень та генеративних можливостей сучасних великих мовних моделей.

За результатами проведеної роботи можна зробити наступні висновки:

1. **Науково-методичний аспект**: Було проаналізовано когнітивні механізми засвоєння інформації, зокрема експоненціальну криву забування Еббінгауза. У ході роботи доведено, що використання класичного підходу є недостатнім для сучасних мобільних додатків, тому було запропоновано модифікацію системи SRS (Spaced Repetition System). Впровадження механізму динамічних вагових коефіцієнтів (Weighted Probability) дозволило перетворити статичний графік повторень на адаптивний процес, де пріоритетність матеріалу для повторення обчислюється індивідуально для кожного слова на основі історії успішних та невдалих спроб користувача.
2. **Технічна та архітектурна реалізація:** Розроблено клієнт-серверну архітектуру, що забезпечує баланс між швидкістю інтерфейсу та потужністю нейромережевих обчислень.

   Android-клієнт: Використання технологій Jetpack Compose та Room Database дозволило реалізувати концепцію "Local-first", забезпечуючи високу чутливість інтерфейсу (Responsive UI) та можливість навчання в офлайн-режимі.

Backend-проксі: Впровадження серверного шару на основі Flask вирішило проблему безпеки (інкапсуляція API-ключів) та дозволило оптимізувати мережеві запити, виконуючи проміжну валідацію відповідей від ШІ, що мінімізує ризик отримання некоректних даних.

3. **Експериментальні дослідження та вибір технологій:** Проведено порівняльний аналіз трьох поколінь мовних моделей (gpt-4.1-nano, gpt-5-mini, gpt-5.4). Емпіричним шляхом встановлено, що модель gpt-4.1-nano є найбільш оптимальною для інтеграції в мобільні системи EdTech. Вона забезпечує 100% валідність словникових одиниць при середньому часі відгуку 11.10 секунд, що є критичним показником для збереження мотивації користувача під час навчальної сесії. Використання більш потужних моделей призводить до необґрунтованого зростання часу очікування, що негативно впливає на загальний користувацький досвід.
4. **Практичне значення та результати:** Розроблене програмне забезпечення продемонструвало стабільність роботи в умовах різного рівня мережевого навантаження. Модульність системи дозволяє легко масштабувати її, інтегрувати нові типи завдань або змінювати постачальника нейромережевих послуг без необхідності переробки основної архітектури додатку. Впровадження інструментів CI/CD та автоматизованих тестів дозволило забезпечити високу якість коду та його легку підтримку.

Перспективи подальшого розвитку: У подальшому планується розширення функціональності системи шляхом впровадження механізмів озвучення, додавання інтерактивних графіків візуалізації прогресу користувача та реалізації можливості синхронізації навчального профілю

між різними пристроями через хмарні сервіси. Також передбачається проведення глибинного аналізу поведінки користувачів для подальшого покращення вагових коефіцієнтів алгоритму адаптивного навчання.

Отже, поставлені завдання виконано в повному обсязі, а розроблений прототип відповідає сучасним вимогам до якості та технологічності програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ebbinghaus, H. (1913). *Memory: A contribution to experimental psychology* (H. A. Ruger & C. E. Bussenius, Trans.). Teachers College, Columbia University. (Original work published 1885)
2. OpenAI. (2024). *OpenAI API documentation*. https://platform.openai.com/docs
3. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 30*. https://arxiv.org/abs/1706.03762
4. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems, 33*. https://arxiv.org/abs/2005.14165
5. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL-HLT*. https://arxiv.org/abs/1810.04805
6. Settles, B., & Meeder, B. (2016). A trainable spaced repetition model for language learning. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. https://doi.org/10.18653/v1/P16-1174
7. Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L., & Sohl-Dickstein, J. (2015). Deep knowledge tracing. *Advances in Neural Information Processing Systems, 28*. https://arxiv.org/abs/1506.05908

8. Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *Proceedings of EMNLP*. https://arxiv.org/abs/1908.10084
9. Google Cloud. (2024). *Vertex AI documentation*. https://cloud.google.com/vertex-ai/docs
10. Amazon Web Services. (2024). *Amazon Bedrock documentation*. https://docs.aws.amazon.com/bedrock/
11. NumPy. NumPy. https://numpy.org
12. Pandas. pandas. https://pandas.pydata.org
13. Pallets. Flask. PyPI. https://pypi.org/project/Flask/
14. McLaughlin, B. (2021). *Programming Kotlin applications: Building mobile and server-side applications with Kotlin.*