Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

# Бакалаврська дипломна робота

на тему:

«Розробка програмного мобільного застосунку для розширення словникового запасу»

Виконав: студент 4 курсу
групи ЗПІ-20б спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки спеціальності)
Степанчук П.В.
(прізвище та ініціали)
Керівник: к.т.н., доцент Романюк О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент: к.т.н., доцент каф. ЗІ Куперштейн Л. М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

**Допущено до захисту**
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О.Н.
(ініціали та прізвище)
« 10 » червня 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
« 12 » березня 2024 р.

**З А В Д А Н Н Я**
**НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ**

Степанчуку Павлу Володимировичу

1. Тема роботи – «Розробка програмного мобільного застосунку для розширення словникового запасу».

Керівник роботи: Романюк Оксана Володимирівна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.

2. Строк подання студентом роботи 3 червня 2024.

3. Вихідні дані до роботи: тип програми – програмний мобільний застосунок; модель розробки – ітеративна; засоби організації даних програми – фреймворки Flutter та Firebase; вхідні дані: мова для вивчення та рівень володіння нею; вихідні дані: створені словники; середовище розробки – Visual Studio Code; мова програмування – Dart; програмне забезпечення для симуляції мобільних пристроїв – Xcode.

4. Зміст текстової частини: вступ; обґрунтування вибору методу розробки та постановка задач дослідження; розробка моделі, методів та алгоритмів роботи програмного продукту; розробка програмного забезпечення; тестування програмного застосунку; висновки; список використаних джерел; додатки.

5. Перелік ілюстративного матеріалу: титульний слайд; актуальність теми розробки; мета, об'єкт та предмет дослідження; задачі дослідження; новизна та практична цінність отриманих результатів; порівняльний аналіз програм-аналогів; модель застосунку; удосконалення методу знаходження слів для вивчення за допомогою використання штучного інтелекту; удосконалення методів практикування та запам'ятовування нових слів; блок-схема алгоритму створення словника; блок-схема алгоритму авторизації та створення профілю;

головна сторінка застосунку; сторінка словника; тестування програми; апробація і публікація бакалаврської дипломної роботи; висновки; фінальний слайд.

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| 1-4 | Романюк О.В., к.т.н., доцент кафедри ПЗ | 13.03.24 | 03.06.24 |

7. Дата видачі завдання 13 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

| Ч.ч. | Назва етапів бакалаврської дипломної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз стану питання та постановка задач дослідження | 14.03.24 – 21.03.2024 | виконано |
| 2 | Розробка моделі застосунку | 22.03.2024 – 25.03.2024 | виконано |
| 3 | Удосконалення методу знаходження слів для вивчення за допомогою використання штучного інтелекту | 27.03.2024 – 01.04.2024 | виконано |
| 4 | Удосконалення методів практикування та запам'ятовування нових слів на основі штучного інтелекту | 02.04.2024 – 07.04.2024 | виконано |
| 5 | Розробка алгоритму створення словника | 08.04.2024 – 14.04.2024 | виконано |
| 6 | Аналіз та вибір засобів для реалізації програмного продукту | 15.04.2024 – 18.04.2024 | виконано |
| 7 | Розробка програмного забезпечення | 19.04.2024 – 09.05.2024 | виконано |
| 8 | Тестування програмного застосунку | 10.05.2024 – 23.05.2024 | виконано |
| 9 | Оформлення матеріалів до захисту БДР | 24.05.2024 – 30.05.24 | виконано |

Студент _____ **Степанчук П.В.**
(підпис) (ініціали та прізвище)

Керівник бакалаврської дипломної роботи _____ **Романюк О.В.**
( підпис ) (ініціали та прізвище)

# АНОТАЦІЯ

Бакалаврська дипломна робота складається з 67 сторінок формату А4, на яких є 35 рисунків, 5 таблиць, список використаних джерел містить 25 найменувань.

У бакалаврській дипломній роботі обґрунтовано актуальність, практичну цінність роботи та її мету, що полягає в удосконаленні процесу розширення словникового запасу за допомогою розробки мобільного застосунку, що сприятиме пришвидшенню процесу пошуку нових слів та ефективному запам’ятовуванню.

Удосконалено метод знаходження слів для вивчення за допомогою використання штучного інтелекту, що допомагає генерувати слова відносно мови для вивчення та рівня користувача. Удосконалено методи для вивчення та запам'ятовування за допомогою генерації завдань відповідно до вибраних користувачем слів, що допоможе ефективніше засвоїти вивчений матеріал.

Для написання застосунку застосовано нову мову програмування Dart та фреймворк Flutter для кросплатформної розробки, що робить можливим написання застосунку одразу під обидві платформи Android та IOS з використанням однієї кодової бази.

Отримані в бакалаврській дипломній роботі результати можуть бути використані для розширення словникового запасу, шляхом вивчення та запам’ятовування нових слів та словосполучень.

Ключові слова: словниковий запас, вивчення слів, іноземні мови, штучний інтелект, мобільний застосунок.

# ABSTRACT

The bachelor's thesis consists of 67 A4 pages, which have 35 figures, 5 tables, the list of sources used contains 25 items.

The relevance, practical value of the work and its purpose were stipulated in the bachelor’s thesis, which is to improve the process of expanding the vocabulary through the development of a mobile application, which will help speed up the process of finding new words and effective memorization.

Improved the method of finding words to learn by using artificial intelligence, which helps generate words relative to the language to learn and the level of the user. Methods for learning and memorization have been improved by generating tasks according to the words selected by the user, which will help to learn the studied material more effectively.

The new Dart programming language and the Flutter framework for crossplatform development are used to write the application, which makes it possible to write the application immediately for both Android and IOS platforms using the same code base.

The results obtained in the bachelor thesis can be used to expand the vocabulary by studying and memorizing new words and phrases.

Key words: vocabulary, word learning, foreign languages, artificial intelligence, mobile application.

ЗМІСТ

# ВСТУП

**Обґрунтування вибору теми дослідження.** Розширений словниковий запас є ключовим елементом в освіті та комунікації, який свідчить про рівень освіченості, ерудиції та культурної розвиненості людини. Він дозволяє точніше виражати свої думки та почуття, сприяє розвитку критичного мислення та аналітичних навичок. Збагачений лексикон розширює можливості сприйняття світу, роблячи інформацію доступною та зрозумілою. Крім того, вміння використовувати різноманітну лексику збільшує успішність в освіті, роботі та особистому житті, сприяючи підвищенню самооцінки та соціальної взаємодії.

Процес розширення словникового запасу є динамічним і постійно вимагає самовдосконалення. Існує кілька шляхів для поповнення лексичного складу мов. Перш за все, активне читання різноманітних текстів, включаючи книги, статті, блоги та інші джерела, допомагає в освоєнні нових слів та виразів. Також корисно вести словниковий щоденник, фіксуючи нові слова та їх контекст [1].

Зазвичай інтерактивне навчання іноземних мов включає в себе використання різноманітних методів і прийомів, спрямованих на активне залучення студентів до процесу вивчення мови. Це можуть бути інтерактивні вправи, використання відео та аудіо матеріалів, групові дискусії та проєкти, інтерактивні ігри, вебінари, використання мобільних додатків та інших сучасних технологій. Такий підхід до навчання сприяє підвищенню мотивації студентів, покращенню їхньої комунікативної компетентності та засвоєнню мови більш ефективним та цікавим способом [2].

Використання інтерактивних методів навчання сприяє активній участі учнів, колективній співпраці, формуванню критичного мислення та навичок розв'язування проблем. Ці методи роблять процес навчання більш цікавим і результативним [3].

Наявні аналоги застосунків розширення словникового запасу не надають достатнього функціоналу.

Тому створення власної програмної реалізації для розширення словникового запасу є актуальним завданням, оскільки воно відкриває більше можливостей для налаштування та адаптації застосунку завдяки розробці з урахуванням конкретних вподобань та стилів вивчення користувачів.

**Зв’язок роботи з науковими програмами, планами, темами.** Робота виконувалася згідно з планом виконання наукових досліджень на кафедрі програмного забезпечення.

**Мета та завдання дослідження**. Метою є удосконалення процесу розширення словникового запасу за допомогою розробки мобільного застосунку, що сприятиме пришвидшенню процесу пошуку нових слів для вивчення та ефективному їх запам’ятовуванню.

Відповідно до поставленої мети потрібно вирішити такі **завдання**:

– здійснити аналіз стану питання та методів розв’язання задачі;

– спроєктувати модель даних мобільного застосунку;

– удосконалити метод знаходження слів для вивчення за допомогою використання штучного інтелекту;

– удосконалити методи практикування та запам’ятовування нових слів на основі штучного інтелекту;

– розробити алгоритм авторизації та створення профілю;

– розробити алгоритм створення словника;

– здійснити програмну реалізацію застосунку;

– провести тестування програмного застосунку;

– розробити інструкцію користувача та описати системні вимоги застосунку для розширення словникового запасу.

**Об’єктом дослідження** є процес розширення словникового запасу за допомогою мобільного застосунку з генерацією слів.

**Предметом дослідження** є методи і засоби розробки програмного забезпечення для розширення словникового запасу для операційних систем Android та IOS.

**Методи дослідження.** У процесі досліджень використовувались такі методи дослідження: порівняльний аналіз функціональних характеристик програм-аналогів для виявлення їх недоліків та постановки задач розробки; теорія алгоритмів для розробки та візуалізації алгоритмів роботи програмного забезпечення; комп'ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

**Новизна отриманих результатів**:

1. Удосконалено метод знаходження слів для вивчення за допомогою використання штучного інтелекту, який на відміну він відомих, дозволяє підбирати нові слова відносно мови для вивчення та рівня користувача, що дозволило створювати індивідуальні набори під кожного користувача та їх потреби.

2. Удосконалено методи запам'ятовування та практикування нових слів, які, на відміну від відомих, використовують штучний інтелект, що дозволяє генерувати завдання окремо для кожного користувача, які базуються на рівні володіння мовою та словах, які вивчаються.

**Практична цінність отриманих результатів.** На основі отриманих теоретичних положень в бакалаврській дипломній роботі запропоновано алгоритми та розроблено програмний мобільний застосунок для розширення словникового запасу.

**Особистий внесок здобувача.** Усі наукові результати, що викладені у бакалаврській дипломній роботі, отримано автором самостійно. У наукових працях, що були опубліковані у співавторстві, автору належать на такі результати: використання хмарних сервісів у мобільній розробці для підвищення продуктивності та функціональності [4]; використання штучного інтелекту у мобільній розробці для підбору слів для вивчення [5].

**Апробація матеріалів бакалаврської дипломної роботи.** Результати роботи доповідалися на:

– LIII Всеукраїнській науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (Вінниця, 2024 р.) [4];

– Молодь в науці: дослідження, проблеми, перспективи (Вінниця, 2024 р.) [5].

**Публікації.** За тематикою дослідження опубліковано 2 наукові праці у збірниках матеріалів конференцій.

# 1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

## 1.1 Аналіз способів розширення словникового запасу

Розширення словникового запасу – це процес збільшення виразів та кількості слів, які людина розуміє і може використовувати в різних життєвих ситуаціях або при письмі. Процес вивчення враховує вивчення нових слів, їх значення або перекладу, правильність вживання залежно від контексту. Розширення словникового запасу допомагає виражати свої думки точніше, вводячи в мову термінологію, яка значно зменшує непорозуміння, розуміти та краще сприймати тексти і ефективніше спілкуватися з іншими людьми.

Розширення словникового запасу можна досягти за допомогою різноманітних методів. Перше, що приходить на думку – виписування нових, незнайомих слів у словник, який може бути представлений у вигляді зошиту, електронної таблиці чи нотаток. Перевага цього методу, те, що людина одразу бачить контекст використання слова  та розвиває писемну пам‘ять записуючи слово в словник. Проте цей метод забирає багато часу, оскільки слово потрібно знайти попередньо щось прочитавши чи почувши. Також записування в словник розвиває тільки пасивну пам’ять [6].

Наступний метод розширення – використання цифрових сервісів задля розширення словникового запасу. Існує велика кількість сервісів, які допомагають цифровізувати метод вивчення. Людина може дивитись фільм з субтитрами і при натисканні на невідоме слово одразу отримує переклад та слово додається в нотатки. Також є сервіси, які мають вже створену програму для вивчення  нових слів. Такий підхід значно економить час, витрачений на записування слів і все зберігається в одному місці. У випадку зі створеними програмами, користувачу не потрібно шукати слова, бо все зроблено за нього. Однак користувач має витратити час на перегляд фільму або ж заздалегідь створена програма може бути доволі обмежена в списку слів і постійно потрібно шукати щось нове.

Одне із сучасних рішень цієї проблеми є автоматизований підбір нових слів та додавання перекладу. Такий метод дозволяє без додаткових зусиль для користувача підібрати нове слово та одразу визначити переклад. Зміст навчальної програми не потрібно оновлювати, оскільки слова генеруються та знаходяться автоматично. З недоліків такого методу є те, що користувач може не до кінця розуміти контекст вживання та не правильно використати слово, однак це можна нівелювати розробивши різні вправи для засвоєння інформації [7]. У поєднанні з мобільним пристроєм такий метод є дуже зручним для щоденного використання тому, що користувач завжди має доступ до телефону і може в будь-який момент відкрити застосунок і дізнатися щось нове або попрактикувати вивчені слова.

Отже, мобільні застосунки для розширення словникового запасу є цінним інструментом для людей, які хочуть зекономити свій час на пошуку нових слів та сконцентруватися саме на отримання нових знань. Такі програми пропонують велику варіативність функцій, які допомагають не витрачати зайвий час на записування чи пошук нових слів, а концентрують увагу користувача саме на засвоєнні матеріалу. У той час як традиційні методи, такі як записування слова в зошит чи в таблицю вже перевірені часом та все ще використовуються багатьма людьми, сучасні інструменти пропонують кращі та ефективніші рішення, які допоможуть користувачам отримати знання за менший час.

### 1.2 Порівняльний аналіз аналогів

Порівняльний аналіз функціоналу аналогів для розширення словникового запасу є важливим етапом при розробці власного продукту, оскільки він дозволяє виявити сильні та слабкі сторони конкуруючих застосунків, оцінити їхні можливості та недоліки. Крім того, аналіз допоможе ідентифікувати нові інноваційні функції або підходи, які можуть зробити продукт більш унікальним та привабливим для цільової аудиторії. Для визначення конкурентних переваг мобільного застосунку розглянемо три популярні мобільні застосунки для розширення словникового запасу:

1. «Quizlet»;
2. «Babbel»;
3. «Anki».

«Quizlet» – це мобільний застосунок, спрямований на розширення словникового запасу шляхом вивчення окремих слів [8]. Основна ідея застосунку полягає у тому, щоб користувач мав єдину бачу зі всіма словами, міг ділитися створеними сетами та міг практикувати нові знання в різних вправах таких як: «Flashcards», «Learn», «Match». Незалежно від уподобань користувача, він може обрати вправу, яка найкраще допоможе засвоїти навчальний матеріал.

Відображення роботи програмного продукту «Quizlet» наведено на рисунку 1.1.

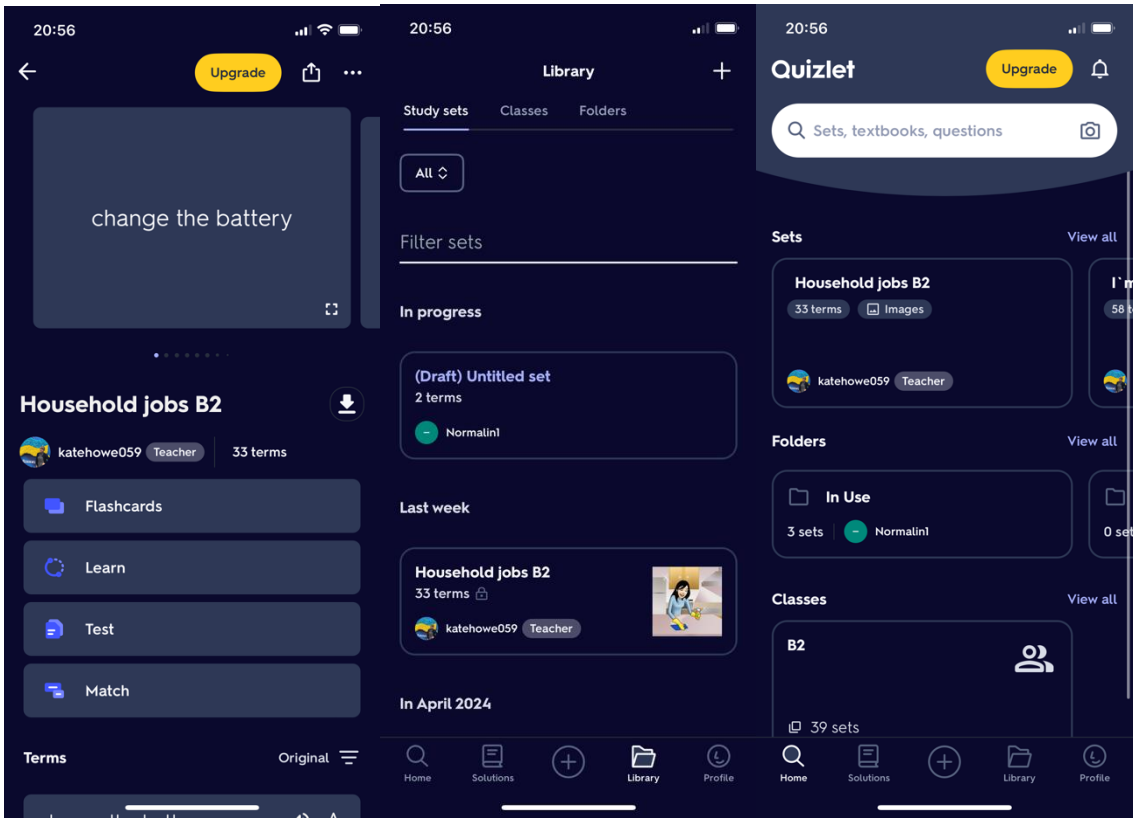



Рисунок 1.1 – Застосунок «Quizlet»

«Babbel» пропонує короткі та ефективні уроки, спеціально розроблені для швидкого вивчення мови [9]. Кожен урок фокусується на певній темі або граматичному аспекті. Застосунок містить різноманітні типи вправ, такі як аудіо-

вправи, вправи на вивчення слів, граматичні завдання тощо, що дозволяє забезпечити різноманітність у навчанні. Також «Babbel» використовує систему повторення для закріплення вивченого матеріалу та підвищення запам’ятовування нових слів та виразів (рисунок 1.2).

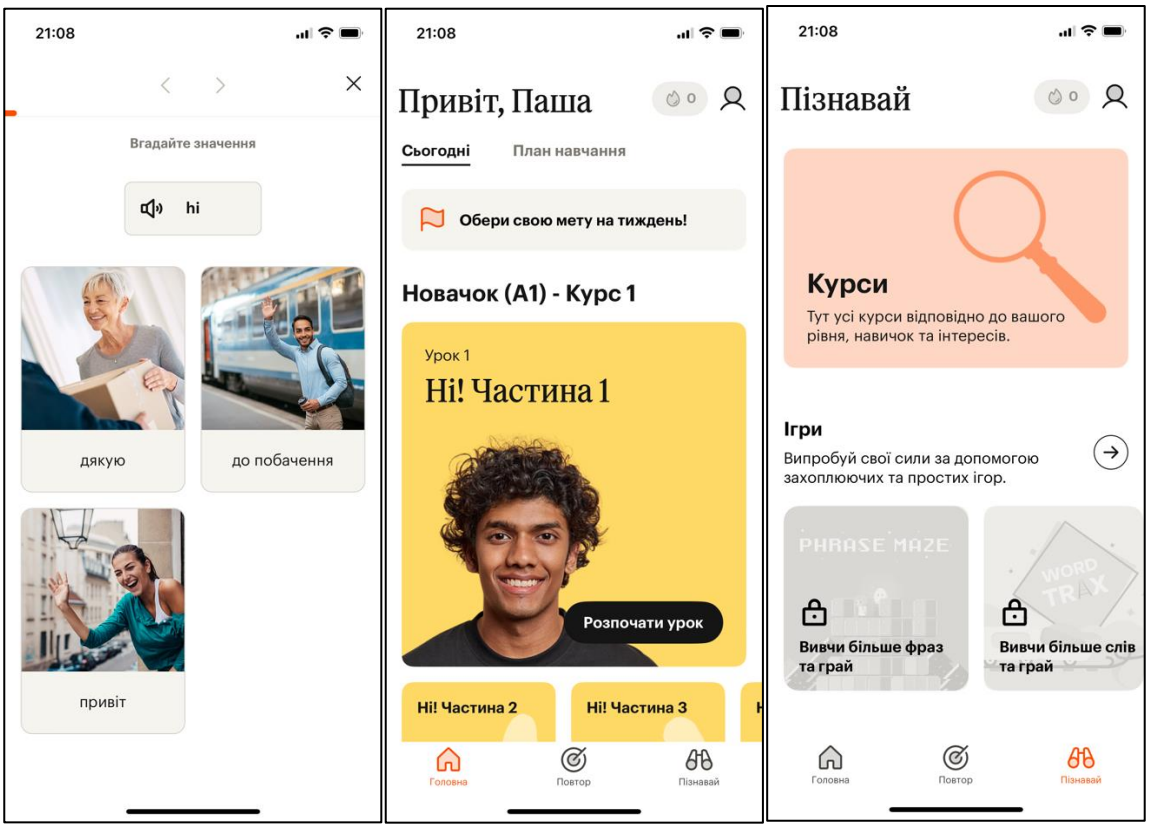



Рисунок 1.2 – Застосунок «Babbel»

«AnkiApp» – це застосунок, який дозволяє користувачам створювати та використовувати набори карток для запам’ятовування різноманітної інформації, включаючи слова, фрази, факти, дати та багато іншого [10]. Основна ідея «Anki» полягає в тому, щоб надати ефективний інструмент для повторення матеріалу за допомогою системи повторень з використанням активного втручання користувача.

«AnkiApp» використовує алгоритм повторень, який адаптується до індивідуальних потреб користувача. Картки, які важко запам’ятати, показуються частіше, а вже відомий матеріал повторюється менш часто (рисунок 1.3).

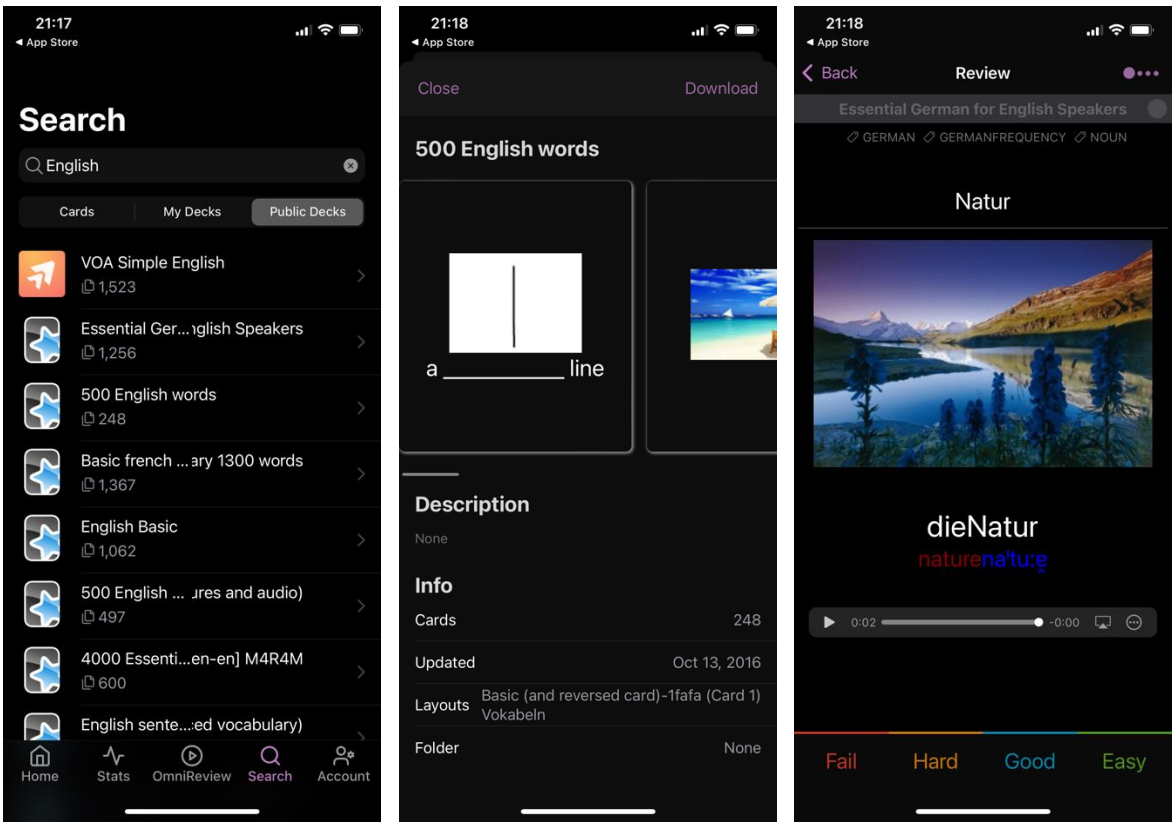



Рисунок 1.3 – Застосунок «AnkiApp»

У результаті розгляду та аналізу аналогів було створено таблицю 1.1 для порівняння їхнього функціоналу із власною мобільною розробкою «JellyVocab».

Таблиця 1.1 – Функціональні характеристики мобільних застосунків

| | Quizlet | Babbel | AnkiApp | JellyVocab |
|---|---|---|---|---|
| Офлайн режим | + | - | + | + |
| Вивчення слів за допомогою написання | + | + | - | + |
| Вивчення слів за допомогою карток | + | - | + | + |
| Вивчення слів у форматі «Вставка в речення» | - | - | - | + |
| Атоматичне генерування слів за допомогою AI | - | - | - | + |
| Сумарний коефіцієнт | 3 | 1 | 2 | 5 |

Нижче наведено опис функцій, за якими проводилося порівняння:

1. Офлайн режим – дозволяє користувачам використовувати застосунок без використання Інтернету.

2. Вивчення слів за допомогою написання – дозволяє користувачам практикувати вивчені слова за допомогою написання слова відносно його перекладу.

3. Вивчення слів за допомогою карток – користувачі можуть вивчати слова за допомогою Flashcards [11], що дозволяє значно прискорити процес запам'ятовування.

4. Вивчення слів у форматі «Вставки в речення» – застосунок генерує речення в якому пропущене слово, а користувач повинен обрати правильну відповідь.

5. Автоматичне генерування слів за допомогою AI – дозволяє користувачам генерувати слово відносно їхнього рівня та одразу отримувати переклад.

Аналізуючи вищенаведені дані «JellyVocab» має значно вищий сумарний коефіцієнт відносно аналогів «Quizlet», «Babbel» та «Anki». Тому розробка мобільного застосунку «JellyVocab» є актуальною, оскільки отриманий продукт відкриває нові можливості для поліпшення користувацького досвіду та розширення можливостей використання програмних засобів для розширення словникового запасу на мобільних пристроях.

Отже, розробка програмних засобів для мобільного застосунку, спрямованого на розширення словникового запасу, є корисною як для бізнес-організацій, так і для звичайних людей. Для бізнесу це може стати ефективним інструментом для покращення комунікації з клієнтами та партнерами у різних середовищах. Також це допомагає працівникам розширити свій словниковий запас для професійних потреб. Для звичайних людей ці застосунки відкривають нові можливості для самостійного навчання мов та підвищення культурного рівня, дозволяючи ефективно вивчати нові слова шляхом виконання різноманітних вправ та тестів у зручний для них час.

### 1.3 Аналіз напрямків удосконалення програмного забезпечення для розширення словникового запасу

Кожна з програм-аналогів має функціонал, який дозволяє презентувати слова для вивчення різними способами, для кращого запам'ятовування та практикування.

Для вивчення слів зазвичай використовуються картки. За допомогою такого методу користувач бачить спереду картки саме слово, а на зворотній стороні – його переклад. Такий підхід спонукає мозок запам’ятовувати і згадувати слово відносно сторони картки. Крім того, картки можна використовувати в обох напрямках: спочатку бачити слово іноземною мовою і пригадувати його переклад, а потім навпаки, бачити переклад і пригадувати оригінальне слово. Це допомагає закріпити знання та зробити процес вивчення більш інтерактивним і ефективним.

Написання слова власноруч – це найефективніший метод для вивчення правопису слова. Цей підхід допомагає активувати моторну пам’ять, що значно покращує запам’ятовування і відтворення інформації. Під час написання слова користувач змушений звертати увагу на кожну букву, що формує правильне уявлення про правопис і структуру слова. Однак, цей підхід має недолік, щоб практикувати написання слова, користувач має його вже досконало запам’ятати, інакше помилки можуть закріпитися в пам’яті. Для удосконалення цього методу програмний засіб  може надавати букви, з яких складається слово і таким чином прискорити запам’ятовування методом написання.

Для того, щоб користувач міг змогу використовувати вивчений матеріал і повсякденному житті, нові слова необхідно практикувати. При розмові дуже часто складно підібрати потрібне слово, яке б з’єднало дві частини речення. З огляду на цю проблему, можливим шляхом її вирішення є розробка функції для практикування, яка емулює таку поведінку надаючи користувачу речення з пропущеним словом.

Деякі люди краще запам’ятовують слово, якщо бачать його в контексті. Такий підхід допомагає створити асоціації та зв'язки між новим словом і його

вживанням у реальних ситуаціях. Тому, для того, щоб удосконалити процес вивчення, можливим шляхом є додавання функції з текстом. Таким чином, користувачі зможуть побачити повну картину використання слова і запам'ятати його швидше маючи приклад застосування.

З огляду на описані вище методи, функціонал застосунків для розширення словникового запасу має декілька різних шляхів для вдосконалення.

### 1.4 Аналіз методів розв’язання поставленої задачі

Для того, щоб розробити мобільний застосунок необхідно спочатку вибрати, на які платформи він має працювати, бо від цього залежить подальший вибір інструментів для реалізації.

Зараз існують дві конкуруючі системи – IOS та Android. Співвідношення користувачі варіюється відносно того, у якій країні вони знаходяться. Якщо розглядати світовий ринок, то Android займає 70% [12]. У США частка користувачів, які користуються IOS – 57.93% [13], тому незважаючи на те, що Android популярніша система, IOS теж займає доволі велику частину ринку. Отже, для того, щоб застосунок міг охопити максимальну частку ринку необхідно розробляти під обидві платформи.

Проаналізуємо різні методи розробки під обидві системи – IOS та Android:

1. Нативна розробка – це метод розробки програмного забезпечення під конкретну платформу. Такий метод зумовлює використання засобів та інструментів, що підтримується цією платформою. Це означає, що програмний засіб пишеться окремо під кожну платформу окремо використовуючи її унікальний набір інструментів. Нативна розробка дає можливість отримати максимальну продуктивність та функціональність за рахунок використання, у зв'язку з тим, що має повний доступ до всіх методів системи та унікальних особливостей платформи. Однак такий підхід більш затратний, якщо порівнювати аналоги, оскільки вимагає створення окремої кодової бази для одного застосунку на різних платформах.

2. Кросплатформена розробка – це метод розробки програмного забезпечення, яке може працювати одночасно на декількох платформах за допомогою одного набору вихідних кодів. Такий метод написання дозволяє розробникам мати лише одну код-базу, що значно пришвидшує розробку та зменшує витрати на створення програмного забезпечення, оскільки пишеться тільки один застосунок, який працює на декількох платформах одночасно. З недоліків такого підходу варто зазначити, що програмний засіб може працювати повільніше та не мати повного набору функцій для конкретної платформи.

3. Змішана розробка (також відома як гібридна) – це метод розробки програмного забезпечення з використанням нативних та кросплатформених інструментів. Такий підхід значно вдосконалює можливості кросплатформеної розробки шляхом надання можливості написання функціоналу за допомогою нативних інструментів. Розробники можуть використовувати один код для різних платформ, зменшуючи час розробки, а також мати можливість використовувати нативні функції для кожної платформи при необхідності. Однак застосунки, які використовують такий метод, дуже важко розширювати та підтримувати, оскільки потрібно писати на різних мовах та в різних місцях, тому з часом код стає заплутаним.

Для розробки програмного мобільного застосунку для розширення словникового запасу було вирішено застосувати крофлатформний метод, тому що такий підхід дає змогу швидше розробити програмний засіб, пишучи один єдиний застосунок під платформи Android та IOS. Крім цього підтримка та вдосконалення мобільного застосунку буде значно ефективніше, оскільки код-база буде одна для обох платформ, що значно зменшить обсяг потрібних ресурсів та пришвидшить процес розробки, порівнюючи з методами-аналогами.

Розглянемо також підходи до організації процесу додавання нових слів для вивчення в мобільних застосунках.

Одним з основних методів є ручне внесення даних, що дозволяє користувачам заносити слова для вивчення власноруч та додавати їх переклад. Потім ці дані використовуються для вивчення та засвоєння матеріалу.

Для того, щоб спростити процес додавання нових слів, деякі застосунку додають можливість одразу визначити переклад для внесеного слова. Тобто, застосунок одразу перекладає слова, яке користувач вписав. Такий підхід доволі зручний, оскільки економить час користувача.

Деякі застосунки, які мають вбудовані словники можуть запропонувати користувачу слово, яке б було цікаве для нього. Використовуючи такий підхід, користувачі можуть повністю сфокусуватися на процесі вивчення і не витрачати час на пошук додаткових нових слів.

У власній реалізації було вирішено втілити вище описані підходи з метою розробки універсального програмного засобу. Також ці методи будуть модернізовані та автоматизовані, щоб дати користувачу максимально ефективно отримувати нові знання.

### 1.5 Впровадження штучного інтелекту в навчанні

Застосування штучного інтелекту (ШІ) в навчанні є актуальною тенденцією сьогодення. Впровадження цієї технології змінює підходи викладання інформації та навчання, що створює нові можливості.

Штучний інтелект має можливість адаптувати навчальні матеріали до рівня знань кожного учня. Системи на основі ШІ можуть аналізувати прогрес навчання студента та рекомендувати завдання, які допоможуть краще засвоїти матеріал. Зазвичай, такі системи не потребують людського втручання і завдання генерують автоматично, що надає системі гнучкості в використанні.

Штучний інтелект може аналізувати великі обсяги даних починаючи з навчальної діяльності студентів, що дозволяє виявляти слабкі місця та прогнозувати успішність, та закінчуючи обробкою новітньої інформації, надаючи можливість оперувати новими даними.

Існують також інтелектуальні навчальні системи, які замінюють або доповнюють викладача або репетитора, надаючи студентам миттєвий зворотній зв’язок і пояснення матеріалу. Штучний інтелект може працювати цілодобово і не потребує відпочинку.

Понад 60% освітніх компаній покладаються на розробку освітніх додатків на основі штучного інтелекту із підтримкою сучасних інструментів і функцій [14]. Такі функції, як багатомовна підтримка, допомагають перекладати інформацію різними мовами, що робить навчання зручним для кожного носія мови.

Отже, ШІ має великий потенціал для покращення навчального процесу, оскільки забезпечує доступ до навчальних ресурсів будь-коли і будь-де. Завдяки аналізу великих даних, ШІ допомагає ідентифікувати найбільш ефективні методи навчання, автоматизує багато рутинних завдань, що дозволяє безпосередньо концентруватися на навчальному процесі. Завдяки правильному застосуванню штучний інтелект може сприяти розвитку та вдосконаленню освітньої системи.

## 1.6 Постановка задач для розробки програмного мобільного застосунку для розширення словникового запасу

Проаналізувавши питання розробки програм для розширення словникового запасу, було визначено перелік задач, які потрібно виконати для розробки мобільного застосунку:

– здійснити аналіз стану питання та методів розв’язання задачі;

– спроєктувати модель даних мобільного застосунку;

– удосконалити метод знаходження слів для вивчення за допомогою використання штучного інтелекту;

– удосконалити методи практикування та запам’ятовування нових слів на основі штучного інтелекту;

– розробити алгоритм авторизації та створення профілю;

– розробити алгоритм створення словника;

– здійснити програмну реалізацію застосунку;

– провести тестування програмного застосунку;

– розробити інструкцію користувача та описати системні вимоги застосунку для розширення словникового запасу.

Технічне завдання на розробку наведено в додатку А.

### 1.7 Висновки

У розділі було досліджено стан питання розширення словникового запасу. Було проведено порівняльний аналіз аналогів Quizlet, Babbel, Anki з розроблювальною програмою. Також було проаналізовано напрямки удосконалення функціоналу застосунку для розширення словникового запасу.

Було розглянуто підходи для створення застосунків для мобільних пристроїв та підходи для реалізації процесу додавання слів для вивчення. Встановлено доцільність застосування кросплатформної розробки та впроваджено методи для кращого запам’ятовування вивченого матеріалу і кращого практикування. Було сформовано задачі, які необхідно досягти при розробці застосунку для розширення словникового запасу.

# 2 РОЗРОБКА МОДЕЛІ, МЕТОДІВ ТА АЛГОРИТМІВ РОБОТИ ПРОГРАМНОГО ПРОДУКТУ

## 2.1 Обробка даних програмного застосунку

Деякі дані можуть бути критичними для правильної роботи програми, тому, задля розробки структури програми необхідно визначити вхідні та вихідні дані для побудови подальшої архітектури, вибору бази даних та написання відповідних методів.

Насамперед, користувач повинен обрати мову, яку він хоче далі вивчати в застосунку. Через те, що більшість мов відрізняються одне від одного і система може підтримувати не всі можливі варіанти, потрібно надати користувачу список для вибору. Цей список має містити назву мови та прапор для кращого та швидшого розуміння. Також користувачу необхідно вибрати рівень володіння вибраною мовою, для того, щоб система підбирала слова відповідної складності.

Задля комфортного використання застосунку, користувач повинен ввести ім'я та завантажити аватар.

Інколи в системі можуть бути несправності, застосунок може оновлюватися з часом, тому необхідно сповіщати юзерів та мати зворотній зв'язок, тому, користувачі за бажанням можуть надати свою електронну пошту.

Для того, щоб користувач міг бачити однакову інформацію на різних пристроях, дані необхідно зберігати в базі даних на сервері. Також, для зручності, дані зберігатимуться локально для швидкого доступу та офлайн використання.

Користувач може використовувати застосунок не тільки для підбору слів, а просто задля вивчення та практикування, тому вхідні дані також міститимуть слова, які юзер введе власноруч.

До вихідних даних належать слова для вивчення. На основі вхідних даних, застосунок прибиратиме відповідні слова з перекладом, що значно поліпшить досвід користування та прискорить процес навчання.

Базуючись на вхідних даних та на отриманих словах, система буде генерувати завдання для подальшого засвоєння та запам’ятовування інформації. Застосунок надасть приклад застосування слова, щоб користувач краще розумів контекст використання та підготує різні приклади, тексти, речення для практикування нових знань

Отже, визначення та обробка вхідних і вихідних даних є критично важливими для розробки ефективної архітектури програми. Правильно налаштована система дозволить користувачам вибирати мову та рівень її володіння, вводити особисті дані, такі як ім’я та аватар, а також надавати зворотний зв'язок через електронну пошту. Зберігання даних на сервері та локально забезпечить доступність і швидкість роботи програми. В результаті, програма зможе підбирати відповідні слова для вивчення, генерувати навчальні завдання та надавати контекстні приклади, що значно покращить процес навчання та користувацький досвід.

### 2.2 Розробка моделі системи

Враховуючи необхідність візуалізації послідовності дій у системі та потребу в зрозумілому інструменті для аналізу робочих процесів, було вирішено розробити UML діаграму прецедентів для моделювання роботи системи. Ця діаграма дозволяє ідентифікувати основні функціональність системи та її взаємодію з користувачами та іншими системами. Вона відображає список усіх можливих дій, які можуть бути виконані користувачами або зовнішніми системами, а також структуру системи та залежності між її компонентами. Такий підхід дозволяє розробникам та зацікавленим сторонам краще зрозуміти функціональність системи та її взаємодію з оточуючим середовищем (рисунок 2.1).

Користувач – це особа або сутність, що використовує застосунок для досягнення своїх цілей або вирішення певних завдань і взаємодіє із застосунком через його інтерфейс, використовуючи різні функції та можливості.

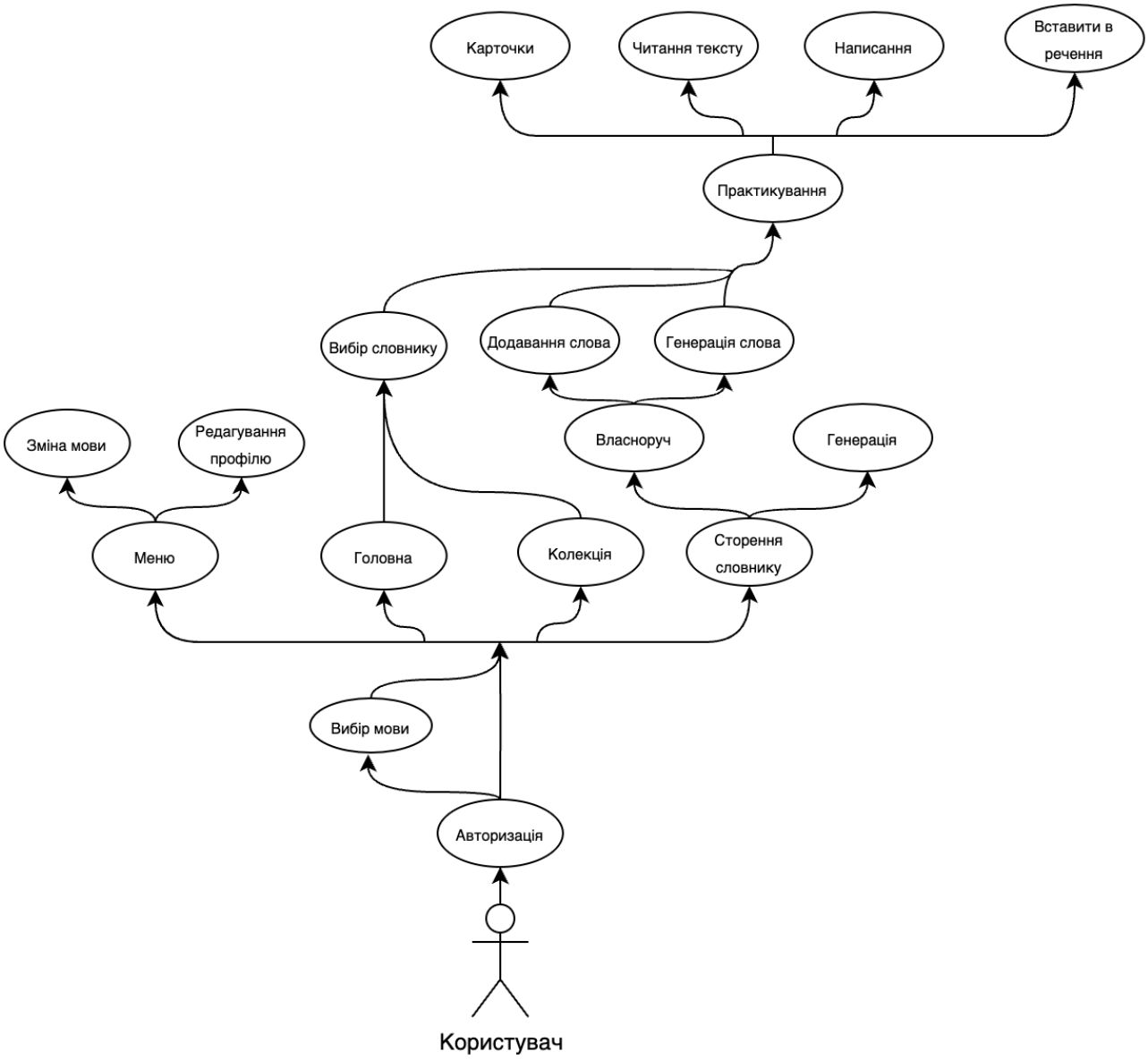



Рисунок 2.1 – Діаграма діяльності мобільного застосунку

Опишемо основні прецеденти користувача:

1) авторизація: користувач відкриває застосунок і використовує авторизацію за допомогою Google або Apple акаунту. Система використовує ці дані, які зберігаються в Firebase, для ідентифікації користувача, після чого користувачу надається доступ на керування застосунком;

2) створення словника: користувач натискає на клавішу для створення словника і потім обирає, яким чином словник буде створений – власноруч або генерацією;

3) генерація: за допомогою штучного інтелекту система генерує словник з випадковими слова відносно рівня користувача;

4) власноруч: користувач переходить до сторінки додавання слів, де він може додати незнайоме слово або згенерувати випадкове;

5) додавання слова: користувач вводить слово та його переклад;

6) практикування: користувач може побачити список слів з словнику та почати практикувати їх за допомогою наступних методів: карточки, читання тексту, написання, тест.

7) карточки: користувач бачить список, який складається з карток. Спереду картка показує переклад, а на зворотному боці саме слово;

8) читання тексту: за допомогою штучного інтелекту система генерує текст, у якому є слова з словнику для читання користувачем;

9) написання: користувач бачить слово і пише власноруч відповідний переклад;

10) тест: користувачу надається речення у якому пропущене слово і список запропонованих слів, одне з яких правильне;

11) колекція: система зберігає і відображає створені користувачем словники в Firebase, на які можна натиснути – і відобразиться сторінка з практикуванням;

12) головна: система відображає словники, які користувач створив, але не практикував, на які можна натиснути і відобразиться сторінка з практикуванням;

13) меню: користувач може вибрати одну з наступних дій: зміна мови, редагування профілю;

14) зміна мови: користувач може обрати мову, яку хочу вивчати, система зберігає цей вибір в Firebase;

15) редагування профілю: користувач може змінити дані про себе (ім’я та свій рівень).

Таким чином, користь UML діаграми полягає в тому, що вона надає структуроване та графічне відображення системи чи програми, що допомагає розуміти її архітектуру та логіку взаємодії між складниками.

Локальне та хмарне сховище відіграють ключову роль у структурі програми, допомагаючи розширювати функціонал застосунку. Вона відповідає за вміст, зберігання всієї інформації застосунку яка стосується користувача, його словників, прогресу навчання.

Хмарне сховище дозволяє синхронізувати зміни даних користувача між різними пристроями. Також у разі видалення застосунку всі дані залишатимуться збережені у хмарі, тому їм буде легко відновити, якщо користувач вирішить повернутися. Локальна база даних допоможе краще контролювати дані, також за допомогою неї можливо реалізувати офлайн режим, тобто дозволяти користувачу використовувати застосунок без з'єднання до Інтернету.

Бази даних можуть бути реалізовані за допомогою різних технології, зберігати різний формат даних. Для розробки застосунку було обрано хмарне сховище Firebase, яке має одразу реалізовану бібліотеку та швидко та ефективно обробляє дані, вона надає можливість додавати велику кількість інформації та доволі легко масштабувати застосунок, оскільки має дуже багато вбудованого функціоналу.

Firebase – це NoSQL [15] база даних, тому для зручного зберігання інформації в локальній базі було обрано бібліотеку Hive. Цей фреймворк дозволяє користувачам працювати з великими обсягами даних, розміщеними в різних джерелах та уникнути конвертування JSON. Також Firebase надає аналітику, яка допомагає розуміти, як користувачі взаємодіють із застосунком або вебсайтом, надаючи важливі дані для прийняття обґрунтованих рішень, що дозволяє оптимізувати досвід користувачів, покращувати продукт та збільшувати його ефективність.

Для зберігання інформації в локальному та хмарному сховищах необхідно розробити схему, яка відображає структурований опис класів і їх взаємозв'язків у програмі. Така схема повинна зображати моделі даних застосунку і забезпечувати зручний формат для зберігання, обробки інформації.

На рисунку 2.2 зображено UML-діаграму класів застосунку для розширення словникового запасу.

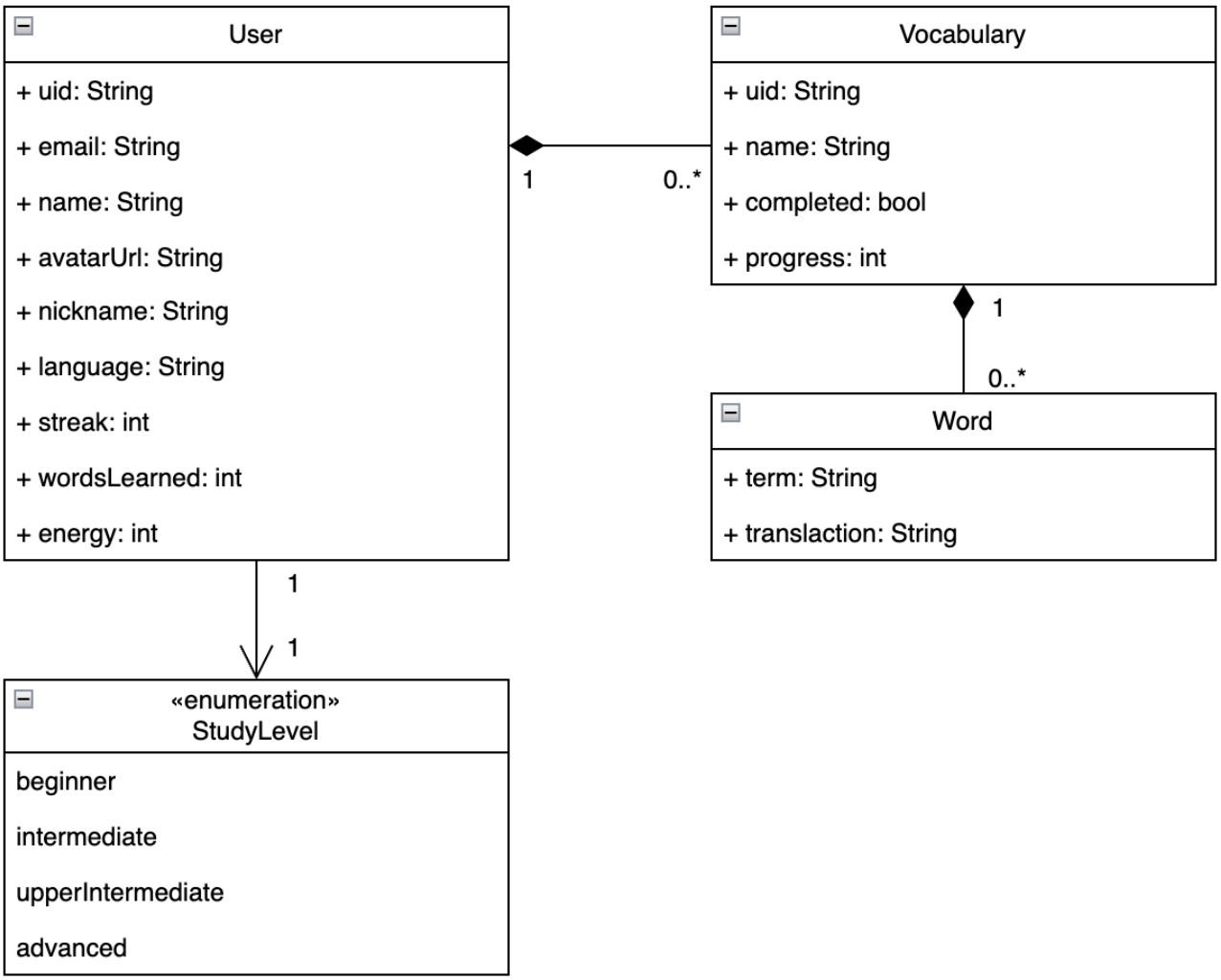



Рисунок 2.2 – Діаграма класів

Клас User має наступні атрибути:

- uid – унікальний ідентифікатор для збереження в базі даних;
- email – електронна адреса;
- name – ім’я користувача;
- avatarUrl – посилання на аватар профілю;
- nickname – ім’я користувача, яке відображатиметься в застосунку;
- language – мова, яку вивчає користувач;
- streak – кількість днів навчання поспіль;
- wordsLearned – кількість вивчених слів;
- energy – енергія (бал).

Клас Vocabulary містить такі атрибути:

- uid – унікальний ідентифікатор для збереження в базі даних;
- name – назва словника;
- completed – чи користувач виконав усі завдання;
- progress – прогрес виконання усіх завдань.

Клас Word містить наступні атрибути:

- term – слово для вивчення;
- translation – переклад слова на рідну мову користувача.

Перелік StudyLevel має такі значення:

- beginner – початковий рівень володіння мовою;
- intermediate – проміжний (середній) рівень володіння мовою;
- upperIntermediate – вище середнього рівень володіння мовою;
- advanced – розширений (високий) рівень володіння мовою.

Класи даних на діаграмі створюють інтерфейс між програмним кодом та базою даних, забезпечуючи зручний та однорідний спосіб взаємодії з даними. Вони представляють собою проміжний рівень абстракції, що дозволяє розробникам працювати з даними у більш зручний та систематизований спосіб, ізолюючі їх від деталей бази даних. Представлені класи не містять ніяких методів, оскільки всі поля є публічними, а розрахунки будуть проводитись в незалежних методах.

Тип зв’язку між сутністю користувач та словник представлену у вигляді композиції «один до багатьох». Використання такого зв’язку полягає в тому, що один користувач може мати велику кількість словників, але останні не можуть існувати, як окремий компонент. Сутність «словник» та «слово» мають такий самий зв’язок – один словник містить багато слів.

Тип зв’язку між сутність користувач та рівень навчання представлений у вигляді асоціації «один до одного», тобто користувач має рівень навчання, який представлений у вигляді enumeration. В подальшому рівень навчання використовуватиметься для генерації слів.

## 2.3 Удосконалення методу знаходження слів для вивчення за допомогою використання штучного інтелекту

Новизна методу знаходження слів для вивчення за допомогою штучного інтелекту полягає в його здатності автоматично і персоналізовано генерувати слова для вивчення, що робить процес розширення словникового запасу більш ефективним і зручним для користувача.

Традиційні методи вивчення нових слів вимагають від користувача самостійного пошуку та вибору слів для вивчення, що може бути трудомістким та не завжди ефективним, оскільки нову інформацію потрібно десь знайти, а це потребує часу. Наприклад, користувач, який хоче розширити словниковий запас, повинен знаходити нові слова дивлячись фільми, читаючи книги або статті, тощо. Якщо користувач знайшов невідоме для себе слово, він має власноруч занести або записати його в застосунок, і тільки з часом буде накопичено список.

Новий метод автоматизує цей процес, використовуючи штучний інтелект для аналізу потреб користувача та підбору найбільш релевантних слів, дозволяючи користувачам зосередитися на вивченні, а не на підготовці.

Метод враховує індивідуальні особливості та потреби користувача, а саме: його рівень знання мови та інтереси.

Таким чином, метод знаходження слів для вивчення за допомогою використання штучного інтелекту є важливим у сфері вивчення мов, оскільки поєднує новітні технології з індивідуальним підходом до навчання, що робить його більш ефективним і зручним для користувачів.

## 2.4 Удосконалення методів практикування та запам’ятовування нових слів за допомогою штучного інтелекту

У сучасному світі, де знання мов є ключовим елементом комунікації та успіху, інноваційні методи вивчення нових слів стають все більш важливими. Штучний інтелект відіграє ключову роль у модернізації та вдосконаленні цих методів, роблячи процес навчання більш дієвим, захоплюючим і

персоналізованим. За допомогою штучного інтелекту можливо удосконалити такі методи:

1. Метод «Флеш-картки». Традиційні флеш-карти вже давно відомі своєю ефективністю у вивченні нових слів. Новація полягає в тому, що ці флеш-карти тепер використовують штучний інтелект (ШІ) для підбору прикладів речень для кожного слова. ШІ аналізує контекст, в якому слово найчастіше використовується, і генерує речення, які допомагають користувачеві краще зрозуміти та запам'ятати слово, роблячи навчання більш контекстуальним і ефективним, оскільки користувачі бачать слова в реальних ситуаціях.

2. Метод «Текст». Суть методу полягає в тому, що штучний інтелект генерує тексти, які включають слова, що вивчаються користувачем. ШІ створює тематичні тексти, які не тільки містять нові слова, але й допомагають користувачам бачити їх у різних контекстах, сприяючи кращому розумінню значення і застосування слів.

3. Метод «Вставка слова». Метод використовує штучний інтелект для генерування речень з пропусками, куди користувач повинен вставляти слова, які він зараз вивчає. Система надає список слів, і користувач повинен вибрати правильне слово для кожного речення. Наприклад, для слова «інновація», ШІ може згенерувати речення: «Нова _____ в технологіях змінила спосіб нашого спілкування». Користувач повинен вибрати правильне слово зі списку, що сприяє активному залученню і глибшому запам'ятовуванню.

Удосконалені методи підвищують ефективність в навчанні, оскільки інтерактивні та контекстуальні підходи роблять процес навчання більш захоплюючим і результативний. Використання ШІ для генерації контенту дозволяє автоматично адаптувати навчальні матеріали під потреби користувача.

Методи враховують індивідуальні потреби та прогрес користувача, що підвищує мотивацію та результативність. ШІ може адаптувати складність завдань і контенту, що дозволяє оптимізувати процес вивчення.

Генерація текстів та речень, що відображають реальні життєві ситуації, допомагає користувачам краще розуміти та запам’ятовувати нові слова, сприяючи практичному застосуванню знань у повсякденному житті.

Ці методи представляють собою новий підхід до вивчення слів, який поєднує інноваційні технології з перевіреними методиками, що робить процес навчання більш ефективним, цікавим і результативним.

### 2.5 Розробка алгоритму авторизації та створення профілю

Для розробки програмних засобів мобільного застосунку розширення словникового запасу необхідно розробити загальний алгоритм авторизації та створення профілю, згідно якого буде працювати застосунок.

Згідно даного алгоритму, зображеного на рисунку 2.3, користувач має увійти в систему, використовуючи Google або Apple акаунт, для подальшого користування.

Якщо під час авторизації сталася якась помилка, система негайно сповістить про це користувача відповідним повідомленням. Помилка може бути викликана поганим Інтернет з’єднанням або примусовим завершення авторизації. У випадку помилки, користувачу буде запропоновано повторити спробу авторизації. Якщо користувач успішно авторизувався, тоді система перевіряє чи він вже має створений акаунт. У випадку відсутності запису в базі даних, користувач повинен вибрати мову, яку хоче вивчати та рівень володіння нею (наприклад, початковий, середній або просунутий). Після вибору мови і рівня, система створює новий акаунт для користувача у базі даних, зберігаючи введену інформацію.

Після успішної авторизації користувача система автоматично перенаправляє його на головну сторінку. На цій сторінці система генерує «слово дня» – нове слово для вивчення, яке оновлюється щодня. Крім цього, головна сторінка відображає інформацію про кількість вивчених слів, прогрес користувача у вивченні мови та його список з його словників.

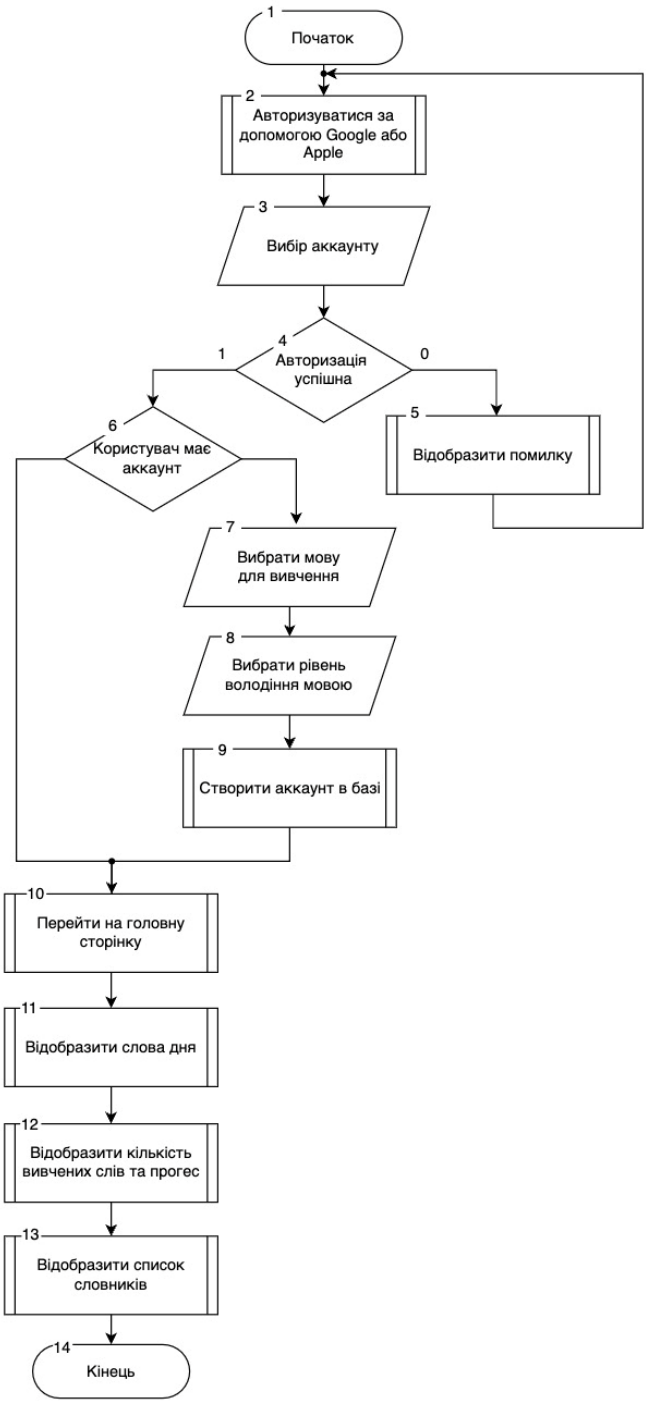



Рисунок 2.3 – Блок-схема алгоритму авторизації та створення профілю

Таким чином, користувач має можливість відстежувати свій прогрес та планувати подальше навчання.

### 2.6 Розробка алгоритму для створення словників з генерацією слів

Для кращого розуміння модуля для створення словників, було розроблено додатковий алгоритм (рисунок 2.4).

Одразу як користувач заходить на сторінку для створення словника, система автоматично створює два пустих записи для заповнення. Це зроблено для того, щоб користувач одразу мав розуміння, що від нього вимагається. Він бачить ці два приклади і може відразу приступити до заповнення власних слів, що спрощує процес створення словника.

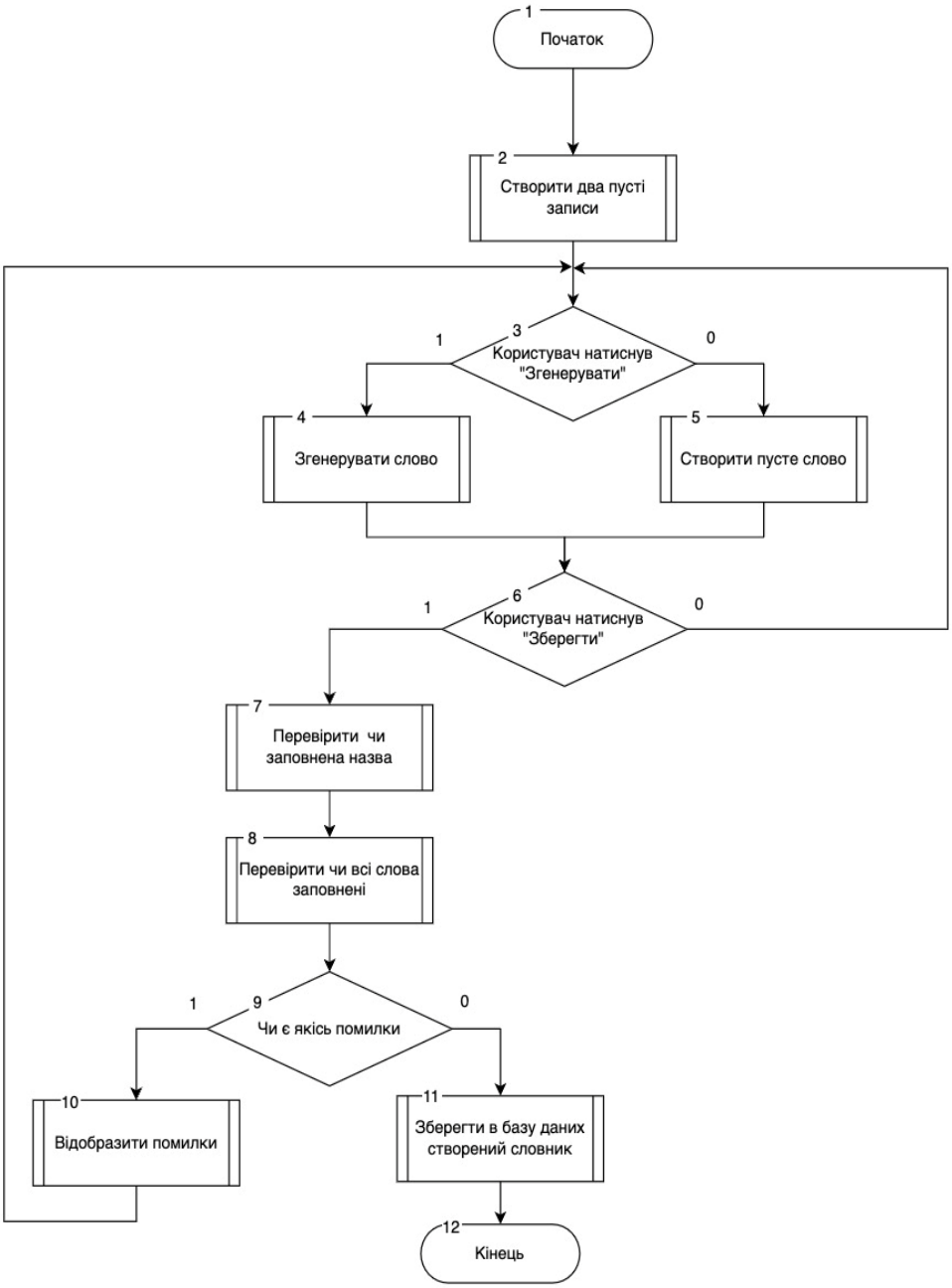



Рисунок 2.4 – Блок-схема алгоритму роботи модуля для створення словників

Якщо користувач натискає кнопку «Згенерувати», система автоматично

створює слово відповідно до рівня та мови користувача. Це дозволяє уникнути затримок, пов’язаних із придумуванням слів, і допомагає користувачу швидше наповнити свій словник. У випадку, якщо користувач не бажає користуватися автоматичним генератором, він має можливість власноруч внести необхідні слова.

Після натискання користувачем кнопки для збереження своїх слів, система проводить перевірку на коректність заповнення всіх видимих полів. Спочатку перевіряється, чи заповнена назва словника, адже без неї неможливо ідентифікувати і зберегти новостворений словник. Потім перевіряється, чи всі поля зі словами не пусті. У випадку виникнення помилки, система одразу відображає її, вказуючи на необхідність заповнення всіх обов'язкових полів, бо без цього словник не буде збережений.  Якщо перевірка полів пройшла успішно, словник зберігається в базі даних.

Створений алгоритм дозволяє автоматизувати систему  створення та управління словниками, забезпечуючи зручний та ефективний процес навчання. Це значно покращує взаємодію користувача із системою, роблячи процес створення словника інтуїтивно зрозумілим і менш трудомістким.

Розглянувши даний алгоритм можна зрозуміти, що розроблений застосунок дає змогу користувачам розширювати свій словниковий запас та практикувати вивчений матеріал.

### 2.7 Висновки

У другому розділі було описано обробку даних програмного застосунку задля визначення вхідних та вихідних даних. Також було розроблено модель системи, що дозволяє проаналізувати робочі процеси та спроєктувати архітектуру. Було визначено місця для удосконалення методу знаходження слів для вивчення та методів практикування та запам’ятовування нових слів. Крім того, було розроблено алгоритми роботи мобільного застосунку,  а саме: алгоритм авторизації та створення профілю, алгоритм для створення словників з генерацією слів.

# 3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

Dart – це оптимізована для клієнта мова розробки швидких програм на будь-якій платформі. Його метою є пропозиція найпродуктивнішої мови програмування для мультиплатформної розробки в поєднанні з гнучкою платформою середовища виконання для фреймворків програм [16].

Перевагами використання мови Dart є:

1. Наявність документації – оскільки Google розробляє інтерпретатор для Dart, усі можливості мови описані детально, що дозволяє швидко отримати відповіді практично на будь-які питання, які можуть виникнути в процесі навчання, або безпосередньо під час написання коду.
2. Висока продуктивність – Dart компілюється в ефективний машинний код,що дозволяє досягти високої продуктивності виконання програм.
3. Гнучкість: Dart підтримує як статичну, так і динамічну типізацію, що дає можливість розробникам обирати найбільш підходящий підхід для своїх проєктів.
4. Оптимізовані бібліотеки і фреймворки – Dart має багату екосистему бібліотек та інструментів, які полегшують розробку. Наприклад, Flutter, фреймворк для розробки мобільних і вебдодатків, використовує Dart як основну мову, що робить його дуже привабливим для розробників, які хочуть створювати кросплатформні застосунки.

Flutter – це кросплатформний фреймворк для створення сучасних, нативних і реактивних програм для IOS і Android [17]. Flutter є проєктом із відкритим вихідним кодом. Flutter використовує Dart, сучасну об’єктно-орієнтовану мову, яка компілюється до рідного коду ARM і готового до виробництва коду JavaScript. Flutter використовує механізм візуалізації Skia 2D, який працює з різними типами апаратних і програмних платформ, а також використовується Google Chrome, Chrome OS, Android, Mozilla Firefox, Firefox

OS та інші. Skia використовує візуалізацію шляху на основі центрального процесора, а також підтримує прискорений сервер OpenGL ES2 [18].

Звісно, що існують також мови-аналоги та фреймворки, які також дозволяють розробляти кросплатформені застосунки.

JavaScript, а саме фреймворк React Native, є потужним інструментом для кросплатформної розробки. React Native дозволяє розробникам створювати мобільні застосунку, які працюють як на платформах iOS, так і на Android, використовуючи єдину кодову базу.

Kotlin Multiplatform (KMP) — це функція мови програмування Kotlin, яка дозволяє розробникам писати код, який можна спільно використовувати на кількох платформах, включаючи iOS, Android, веб-додатки та програми для ПК. Kotlin, розроблений JetBrains [19], відомий своїм лаконічним синтаксисом і сучасними функціями.  У таблиці 3.1 наведено порівняння мов-аналогів.

Таблиця 3.1 – Порівняння мов програмування та їх фреймворків

| Мова (Фреймворк) | JavaScript (React Native) | Kotlin (KMP) | Dart (Flutter) |
|---|---|---|---|
| Одна кодова база для UI | + | - | + |
| Офіційна підтримка від Google | - | - | + |
| Можливість розробляти веб-сайти | - | + | + |
| Оптимізація скомпільованого UI | - | + | + |
| Сумарний коефіцієнт | 1 | 2 | 4 |

Аналізуючи таблицю порівняння мов програмування, Dart має найбільший коефіцієнт, тому найкраще підходить для розробки.

Firebase забезпечує широкий спектр функціональності, включаючи базу даних у реальному часі, аутентифікацію користувачів, хостинг, зберігання файлів, аналітику, моніторинг продуктивності та інше. Вона розроблена з

урахуванням потреб розробників у простоті використання та ефективності, що робить її популярним вибором для розробки різноманітних застосунків [20]. Крім того, Firebase пропонує докладні аналітичні звіти, що дозволяють розробникам зрозуміти поведінку користувачів та поліпшити додатки на основі зібраних даних.

Для того, щоб зручно розробляти застосунки та використовувати повний спектр запропонованих мов, необхідно визначити середовище розробки. Якщо говорити про Dart то зазвичай застосовують Android Studio або VsCode, оскільки вони мають офіційну підтримку мови. У таблиці 3.2 наведено порівняльний аналіз двох середовищ розробки.

Таблиця 3.2 – Порівняння середовища розробки

| Середовище розробки | Android Studio | VsCode |
|---|---|---|
| Підтримка Dart сніпетів через розширення | - | + |
| Рефакторинг коду застосовуючи dart lint | - | + |
| Підтримка розширення для бібліотеки bloc | - | + |
| Загальний коефіцієнт | 0 | 3 |

Оскільки VsCode має більший бал через більшу підтримку розширень, саме це середовище розробки було обране для застосування.

Отже, Flutter є фреймворком, який використовує Dart як мову програмування. Dart використовується для написання додаткового коду для реалізації бізнес-логіки та взаємодії зі складовими користувацького інтерфейсу, а також для опису власних віджетів у Flutter. У поєднанні з Firebase, Flutter отримує додаткові переваги в розробці мобільних застосунків, враховуючи реальний час оновлень, автентифікацію, зберігання даних та іншими функціональними можливостями, працюючи в одному інтегрованому середовищі. VsCode є кращим середовищем розробки, оскільки надає більше розширень для роботи з Dart та Flutter.

### 3.2 Розробка модуля для створення і генерації словника

Головною метою створення застосунку є створення і генерації словника. Для того, щоб користувач мав змогу вводити свої слова та додавати нові, відбувається ініціалізація BLoC полів, що допоможе легко валідувати необхідні поля та масштабувати застосунок [21]. Цей підхід дозволяє реалізувати ефективну комунікацію між компонентами і забезпечує можливість динамічної зміни даних та їх відображення на сторінці. Крім того, ініціалізація цих полів дозволяє використовувати розширені можливості управління станом додатку, такі як асинхронні операції та обробка помилок. Це сприяє покращенню користувацького досвіду та забезпечує більш гнучку архітектуру додатку, яка легко підлаштовується під змінні потреби користувачів. На рисунку 3.1 зображено сторінку для створення словників.

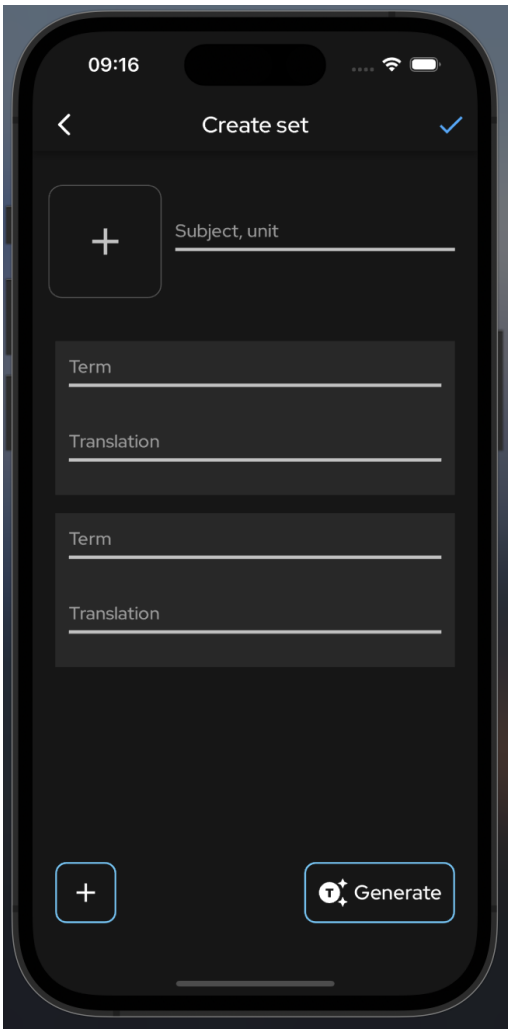



Рисунок 3.1 – Ініціалізація полів

Для того, щоб користувач розумів, що на сторінці створення йому потрібно робити, система автоматично додає два пусті записи для подальшого заповнення користувачем.

Кожне слово в словнику складається з двох значень: терміну та його перекладу. Для кращого управління списком слів та показування їх на UI, поля терміну та перекладу були згруповані. Це дозволяє зробити інтерфейс більш зрозумілим та зручним для користувача, оскільки вони представлені як єдиний об’єкт, спрощуючи введення та редагування інформації [22]. Таке групування також полегшує роботу системі зі словником, оскільки дозволяє однаково обробляти термін та його переклад, спрощуючи логіку взаємодії з даними.

На рисунку 3.2 зображено групи з полями терміну та перекладу.

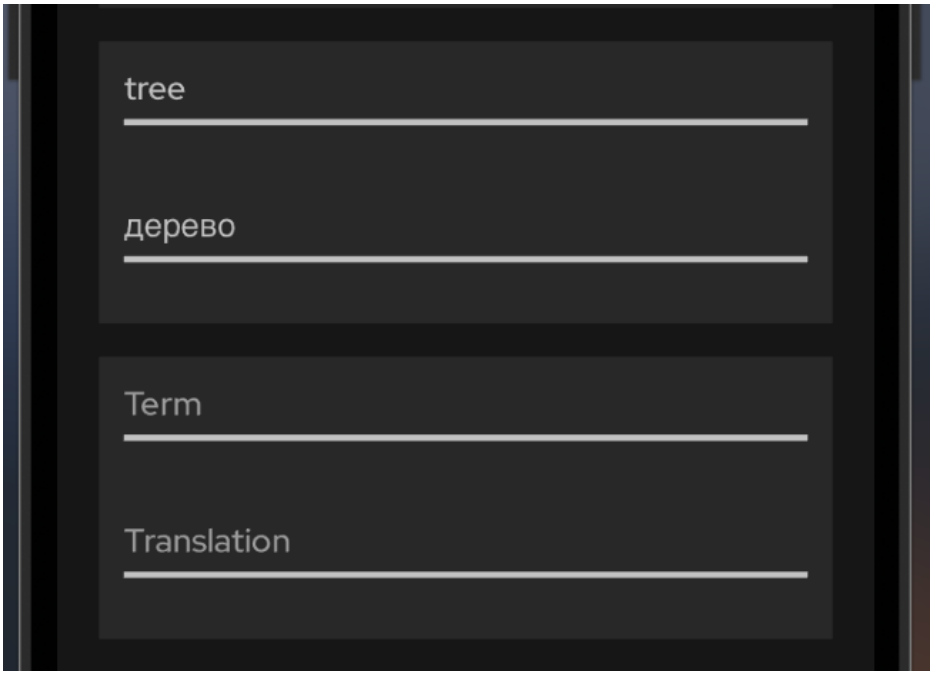



Рисунок 3.2 – Групи з полями терміну та перекладу

Одна з головних функцій на цій сторінці – генерування слів відносно рівня користувача. Коли користувач натискає на кнопку, яка відповідає за генерацію на екрані відразу створюється група, яка відображає завантаження. Водночас система перевіряє чи має користувач не заповнену групу, якщо так, тоді запис заміняється згенерованим. Це зроблено для того, щоб покращити досвід

взаємодії користувача, щоб уникнути додатковий та зайвих натискань. Генерація терміну та перекладу відбувається за допомогою сервісу, який отримує рівень володіння мовою та саму мову, яку вивчає користувач. Якщо сталась якась помилка, тоді система припиняє генерацію, видаляє створену групу та відображає помилку. На рисунку 3.3 завантаження групи.

Рисунок 3.3 – Завантаження групи

Після того, як користувач додав або згенерував усі необхідні слова, він має зберегти свій словник, для того, щоб потім вивчати та практикуватись. Після натискання на збереження словнику, система починає валідувати всі введені дані. Якщо якесь з полів не заповнено, тоді система припиняє зберігання та відображає помилку.

Після процесу валідації, якщо не було виявлено помилок, система зберігає створений користувачем словник в базі даних.

Таким чином, було розроблено модуль для створення і генерації словника, який дозволяє користувачам не лише створювати власні словники, але й автоматично генерувати нові слова відповідно до їх обраного рівня володіння мови. Такий підхід сприяє зручності та ефективності процесу вивчення мови, роблячи навчання більш доступним та цікавим для користувачів.

### 3.3 Розробка сервісу для реєстрації користувача

Згідно з поставленими завданнями, система має генерувати слова з урахуванням обраної мови та рівня володіння нею. Для цього користувач повинен вибрати відповідні налаштування, такі як мова і рівень складності. Під час авторизації, система перевіряє наявність вже створеного акаунту.

На рисунку 3.4 зображено сторінку авторизації.

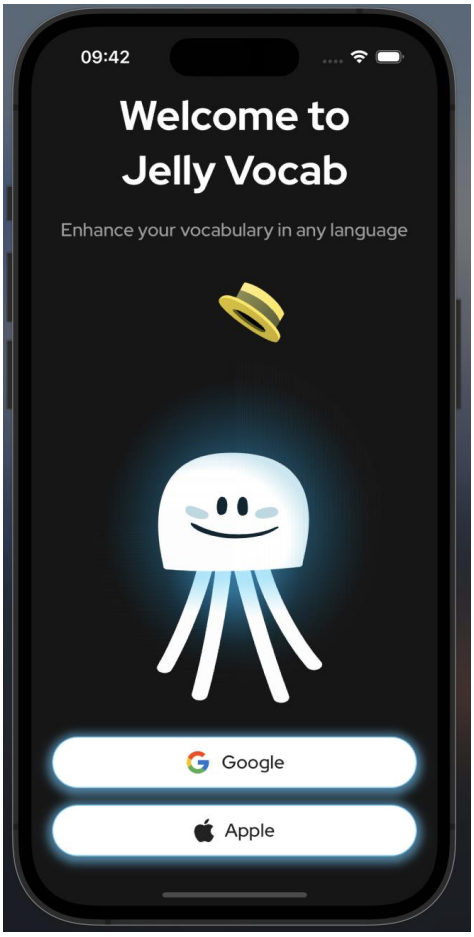



Рисунок 3.4 – Сторінка авторизації

Також на сторінці авторизації користувач бачить векторну анімацію медузи, що є основним персонажем мобільного застосунку. Елементи розробленої анімації медузи наведені на рисунку 3.5

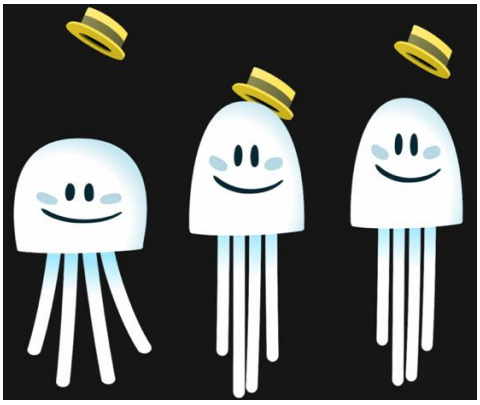


Рисунок 3.5 – Елементи векторної анімації медузи

Якщо система виявила відсутність облікового запису користувача, вона автоматично перенаправляє його на спеціальний екран, де необхідно здійснити вибір мови та визначити рівень володіння нею. Цей процес розпочинається з ініціалізації усіх необхідних полів, щоб користувач міг зручно і швидко здійснити необхідні налаштування.

На рисунку 3.6 зображена сторінка для вибору мову та визначення рівню володіння нею.

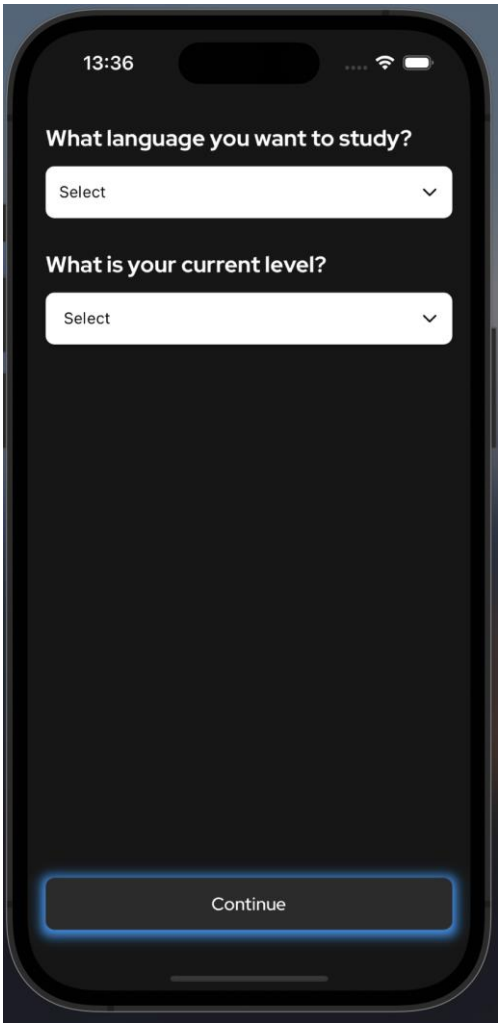



Рисунок 3.6 – Сторінка для вибору мови та визначення рівня володіння

Після натискання на вибір мови, на екрані з’являється сторінка зі списком мов. Користувачі можуть вибрати мову для вивчення, які підтримує застосунок. На рисунку 3.7 зображено сторінку зі списком мов.

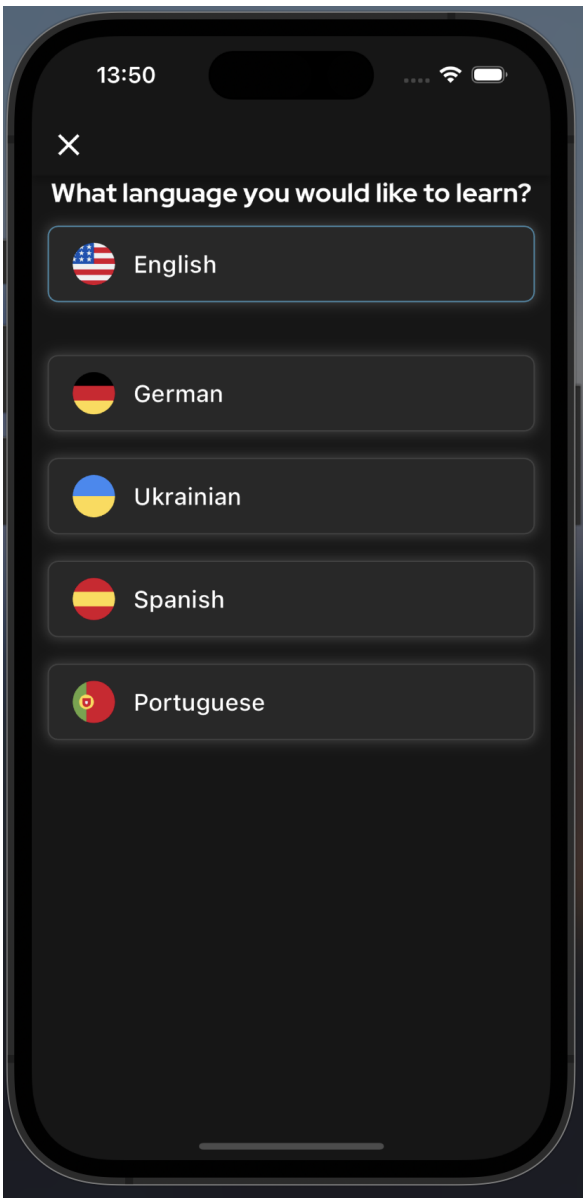



Рисунок 3.7 – Сторінка для вибору мови для вивчення

Після того, як користувач вибирає необхідні налаштування та натискає кнопку для збереження, система перевіряє введені дані на вірність. Якщо дані коректні, вона починає процес збереження та створення облікового запису. Під час цього процесу система може відобразити користувачеві підтвердження успішного створення акаунту або повідомлення про можливі проблеми з введеними даними, які потребують уваги користувача для виправлення. Після успішного створення облікового запису користувач зможе увійти в систему та використовувати її функціонал.

На рисунку 3.8 зображена сторінка для створення профілю.

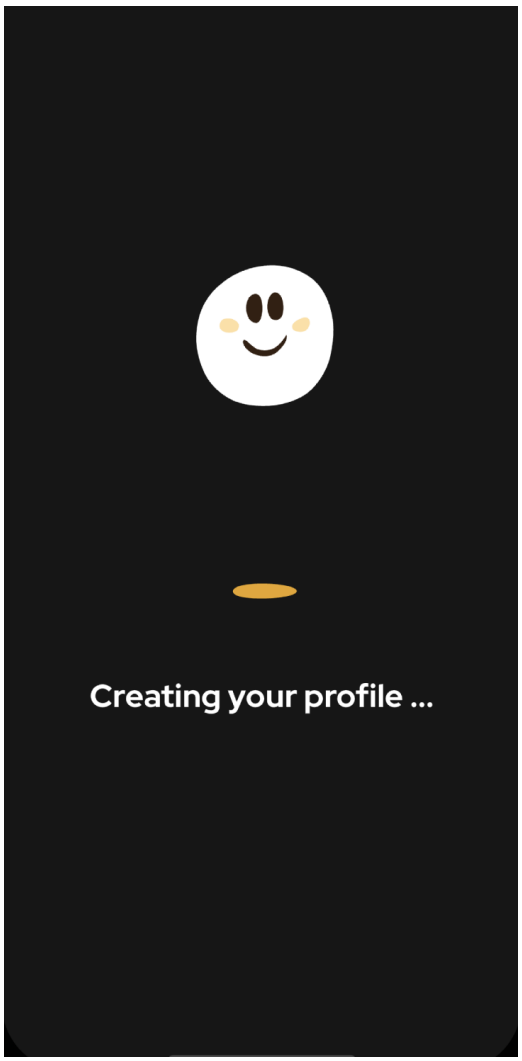



Рисунок 3.8 – Сторінка очікування створення профілю

Отже, сервіс для реєстрації користувача дозволяє зареєструватися в системі за допомогою облікового запису Google або Apple, роблячи процес авторизації швидше і зручніше та уникаючи необхідності запам’ятовувати нові паролі, спрощуючи доступ користувачів до застосунку.

### 3.4 Розробка методів запам’ятовування та швидкого засвоєння нових слів

Для того, щоб користувач міг не тільки знаходити нові слова, а й запам’ятовувати їх необхідно розробити методи для швидшого засвоєння нового матеріалу.

Перший метод «Текст» – це практикування слів в контексті. Користувачу буде легше запам’ятати слова, якщо він одразу бачить приклади вживання в

повноцінному тексті. Для того, щоб не шукати потрібний текст, який містив би усі слова, які користувач вибрав для вивчення, оскільки існує дуже велика кількість слів і, відповідно, ще більша кількість комбінацій, тексти повинні генеруватися автоматично. Для цього завдання було застосовано штучний інтелект, бо чудово впорається з поставленою задачею.

На рисунку 3.9 зображено сторінку з методом «Текст».

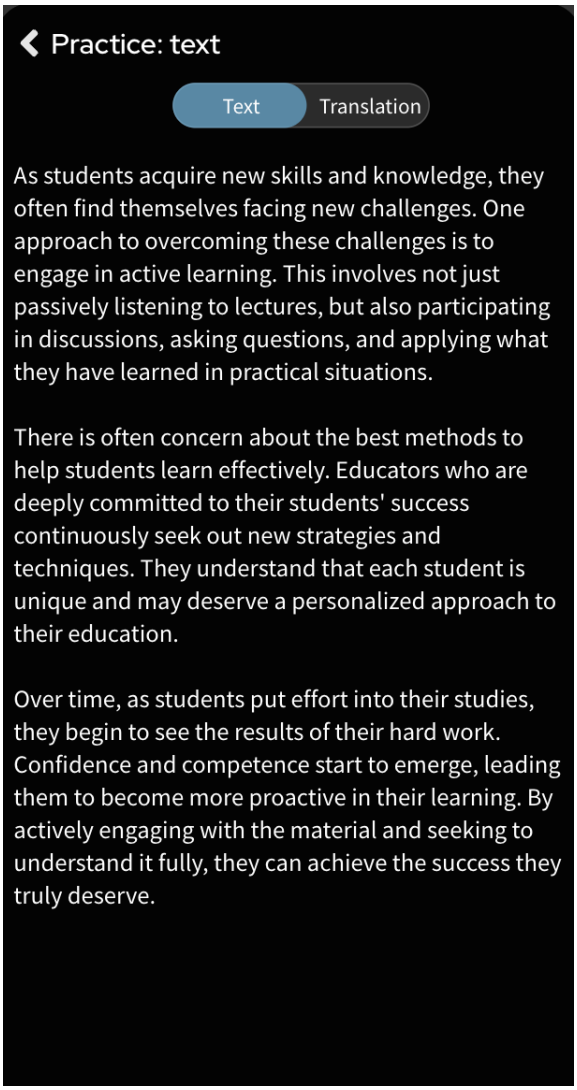



Рисунок 3.9 – Метод «Текст»

Для того, щоб у користувача не було жодних проблем з розумінням тесту, цей метод також передбачає повний переклад усіх слів.

Другий метод «Вставка слова» – призначений для перевірки засвоєння вивченого матеріалу та вміння застосовувати нові знання. Люди, які не

повноцінно володіють мовою, зазвичай спочатку формують думку на рідній мові і вже далі перекладають сформовані речення. Тому, для того, щоб користувач міг застосовувати вивчені слова, було вирішено розробити метод, який показує речення, у якому відсутнє слово. Щоб юзер швидше і впевненіше практикував цей метод система показує список з 4 слів, які користувач вивчав, але тільки одне слово є правильним.

На рисунку 3.10 зображено метод «Вставка слова».

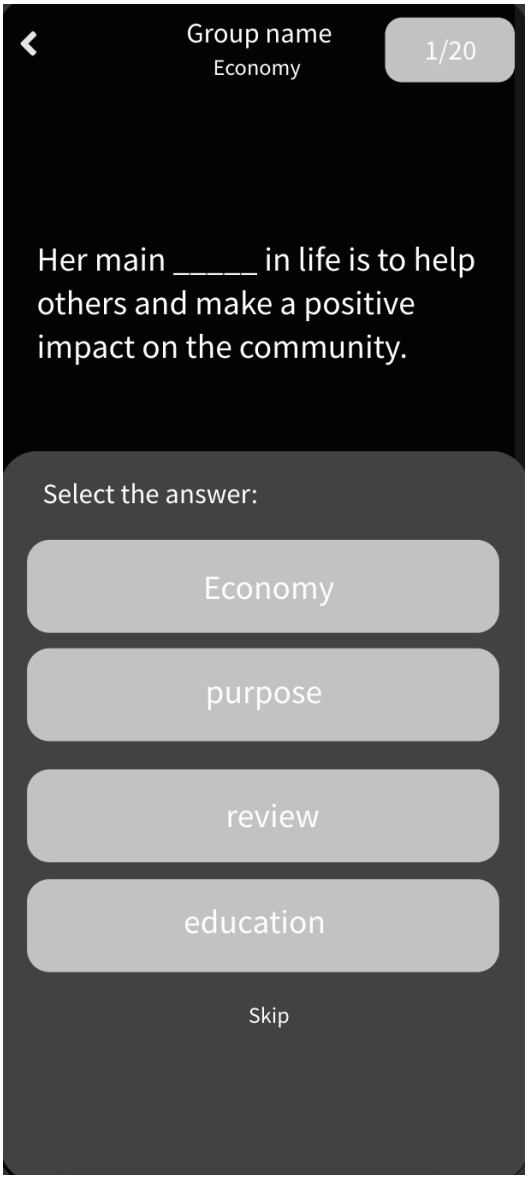



Рисунок 3.10 – Метод «Вставка слова»

Для поліпшення процесу запам’ятовування було розроблено метод з назвою «Flashcards». Цей метод містить в собі звичайні картики для вивчення,

тобто з одної сторони слово, а з іншої – переклад. Але задля прискорення згадування та швидкості запам’ятовування в карти також було додано ще приклад застосування. Користувач може згадати слово, виходячи з контексту, і одразу бачити, як і в якій формі воно застосовується. На рисунку 3.11 зображено метод «Flashcards».

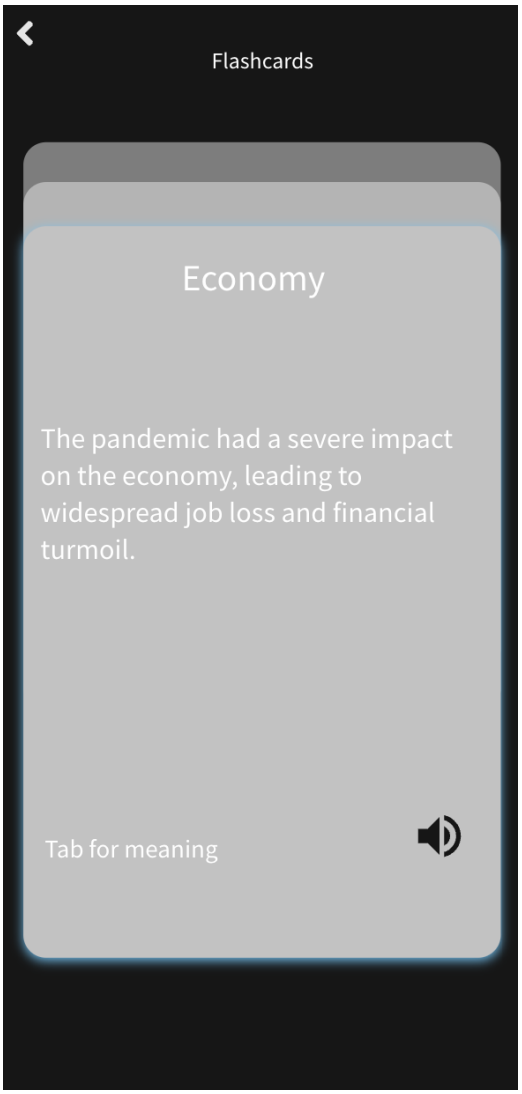



Рисунок 3.11 – Метод «Flashcards»

Слова, які користувач буде вивчати дуже складно передбачити і підготувати вже готові речення або приклади, оскільки їх може бути безліч. Тому, задля розробки методів «Flashcards» та «Вставка слова» було використано штучний інтелект, який буде генерувати речення з пропущеним словом та приклади застосування слова і перевіряти правильність виконання завдань.

### 3.5 Висновки

У третьому розділі було подано обґрунтовано вибір засобів реалізації програмного продукту, технологій розробки та модулів. Під час обгрунтування вибору засобів реалізації програмного продукту було визначено, що використання мови програмування Dart спільно з фреймворком Flutter та хмарним сервісом Firebase від Google є оптимальним рішенням для розробки мобільної системи.

# 4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАСТОСУНКУ

## 4.1 Аналіз методів тестування програмного забезпечення

Важливим етапом розробки програмного застосунку є тестування створеного програмного продукту. Основною метою тестування програмного забезпечення є виявлення дефектів або помилок у програмному застосунку та забезпечення його правильної роботи перед випуском для користувачів.

Flutter-тестування включає в себе різноманітні методи, спрямовані на перевірку правильності та ефективності програмних продуктів. Серед них варто виокремити такі:

- інтеграційне тестування;
- золоті тести;
- мокінг і ін’єкцію залежностей.

Розглянемо детальніше кожний із них.

Інтеграційні тести зосереджені на перевірці взаємодії та інтеграції між різними компонентами або модулями в програмі [23]. Ці тести перевіряють поведінку програми в цілому, включаючи навігацію, потік даних і взаємодію між різними екранами або функціями. Тобто, інтеграційні тести допомагають забезпечити правильну роботу всієї системи як єдиної сутності. Flutter надає пакет «flutter_driver» для написання інтеграційних тестів, які взаємодіють із програмою через окремий процес.

Золоті тести використовуються для забезпечення узгодженого відтворення компонентів інтерфейсу користувача на різних платформах і конфігураціях пристроїв [24]. У золотих тестах знімки екрана компонентів інтерфейсу користувача робляться під час виконання тесту та порівнюються з еталонними зображеннями (золотими файлами), щоб виявити будь-які візуальні регресії. Крім того, золоті тести можуть слугувати важливим ресурсом для визначення якості коду та виявлення потенційних проблем у розробці програми.

Мокінг і ін’єкція залежностей допомагають ізолювати код програми від зовнішніх залежностей, таких як API або бази даних, що дозволяє забезпечити

передбачувану роботу системи та спростити процес тестування [24]. Фреймворки впровадження залежностей, як «get_it» або «provider», можна використовувати для керування залежностями та полегшення моделювання в тестах.

### 4.2 Тестування мобільного застосунку розширення словникового запасу

Тестування продуктивності є важливим елементом перевірки ефективності та надійності мобільного застосунку, яке дозволяє оцінити, як швидко та ефективно працює застосунок при реальному навантаженні користувачів. Результат тестування наведено на рисунку 4.1.

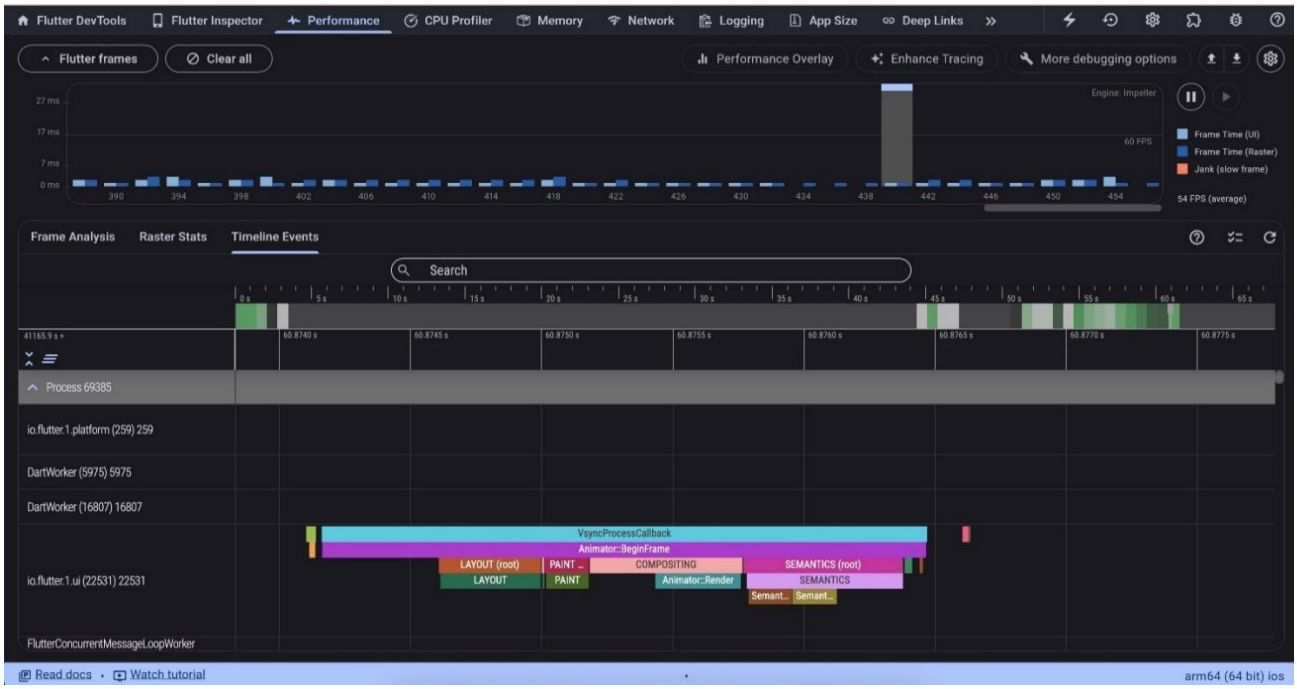



Рисунок 4.1 – Тестування продуктивності

Перевірка створення словника з пустими даними є одним з ключових тестувань розробленого застосунку. У разі спроби збереження створеного словника без назви або слів з їх перекладами, система повідомляє про помилку та пропонує ввести дані. Результат тестування зображено на рисунку 4.2.

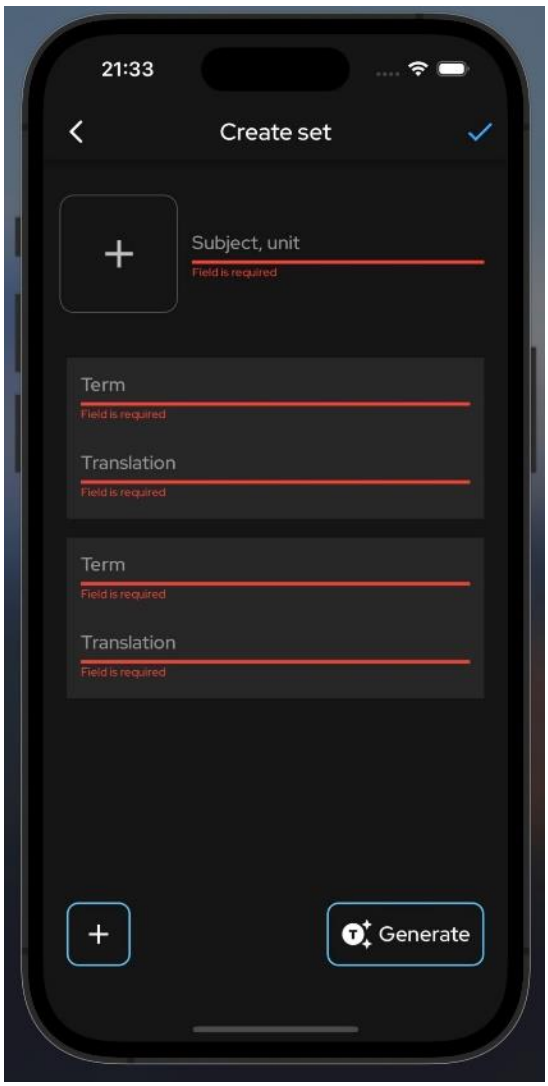



Рисунок 4.2 – Тестування створення словника з пустими даними

Під час редагування профілю та зміни адреси електронної пошти, користувач може випадково ввести неправильний формат пошти. Система повинна провести перевірку введеної адреси електронної пошти на відповідність стандартному формату. Якщо введений формат не відповідає очікуваному, система повинна відобразити відповідне повідомлення про помилку та попросити користувача ввести коректну адресу електронної пошти. Таким чином, забезпечити правильність та цілісність введених даних у профілі користувача.

На рисунку 4.3 зображено результат тестування на правильність введення електронної адреси.

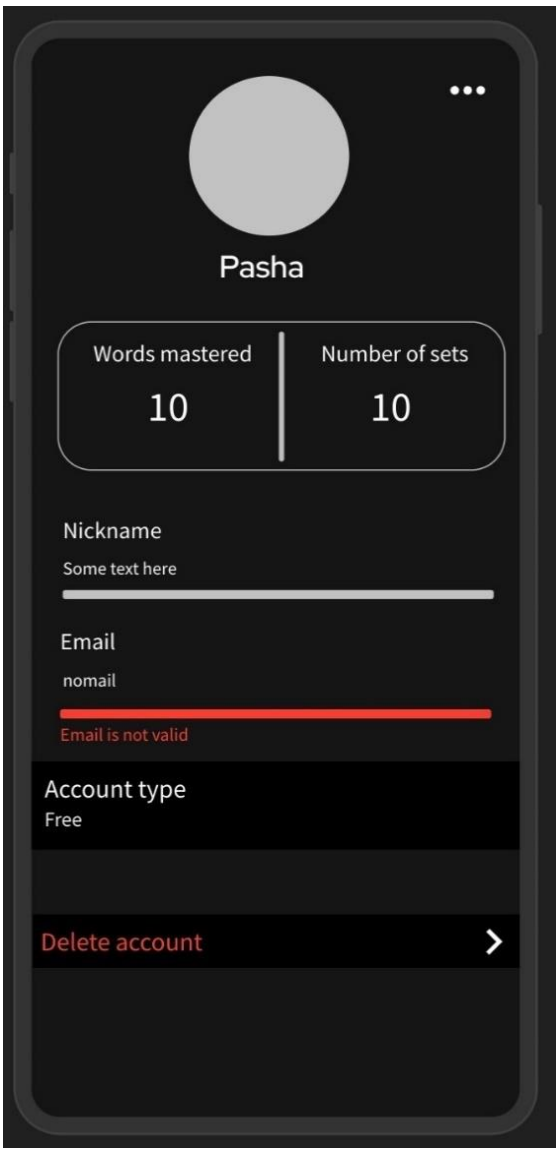



Рисунок 4.3 – Тестування правильності електронної адреси

Під час вивчення слів методом «Вставки пропущеного слова», користувач може вибрати неправильну відповідь, оскільки він знаходиться у процесі навчання і може ще не мати достатнього володіння даної мови. У такому випадку система повинна надати відповідне повідомлення про помилку та підказати користувачеві правильну відповідь. Це допоможе користувачу зрозуміти свою помилку і навчитися правильному варіанту, сприяючи його подальшому навчанню.

На рисунку 4.4 зображено результат тестування відображення помилки при виборі неправильної відповіді.

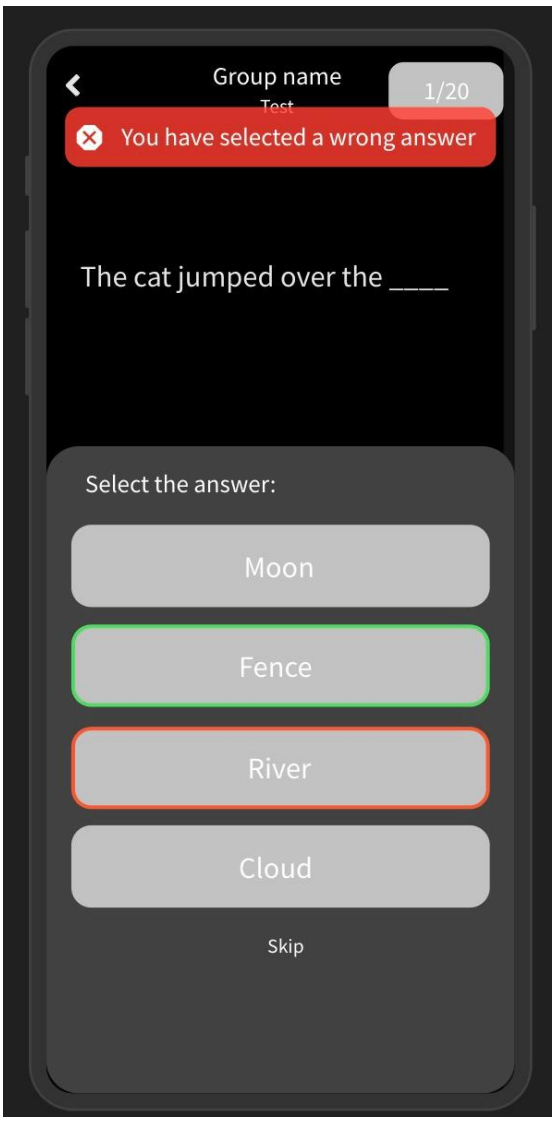



Рисунок 4.4 – Тестування неправильно обраної відповіді

Під час вивчення слів методом «Написання» користувач може написати неправильний переклад слова, що є допустимим. У такому випадку система має повідомити, що введена відповідь не є правильною і потрібно спробувати ще. Якщо користувач, все-таки, не може згадати слово, тоді в нього є опція підказки. Якщо користувач навіть після надання підказки не може згадати слово, він має можливість пропустити його і перейти до наступного слова в навчальному процесі. Це дозволяє уникнути зациклення на одному слові і продовжити ефективне вивчення іншого матеріалу.

На рисунку 4.5 зображено результат тестування відображення помилки при вводі неправильного перекладу.


Рисунок 4.5 – Тестування неправильно введеної відповіді

Отже, тестування системи включає в себе перевірку функціональності, надійності та продуктивності програмного забезпечення. Здійснене тестування включало проведення інтеграційного тестування для перевірки взаємодії між різними компонентами системи, золотих тестів для перевірки найважливіших функцій та сценаріїв, а також мокінгу і ін’єкції залежностей для імітації певних умов і динамічних взаємодій.

### 4.3 Розробка інструкції користувача

Після авторизації користувач переходить на головну сторінку, яка відкривається за замовчуванням.

Зображення головного екрану показано на рисунку 4.6.

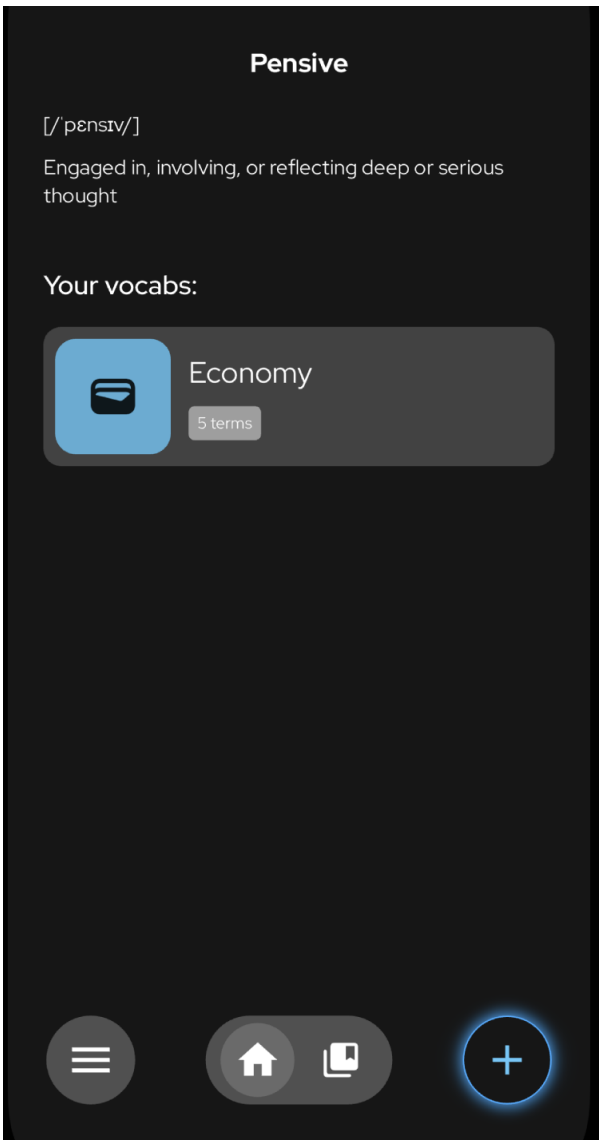


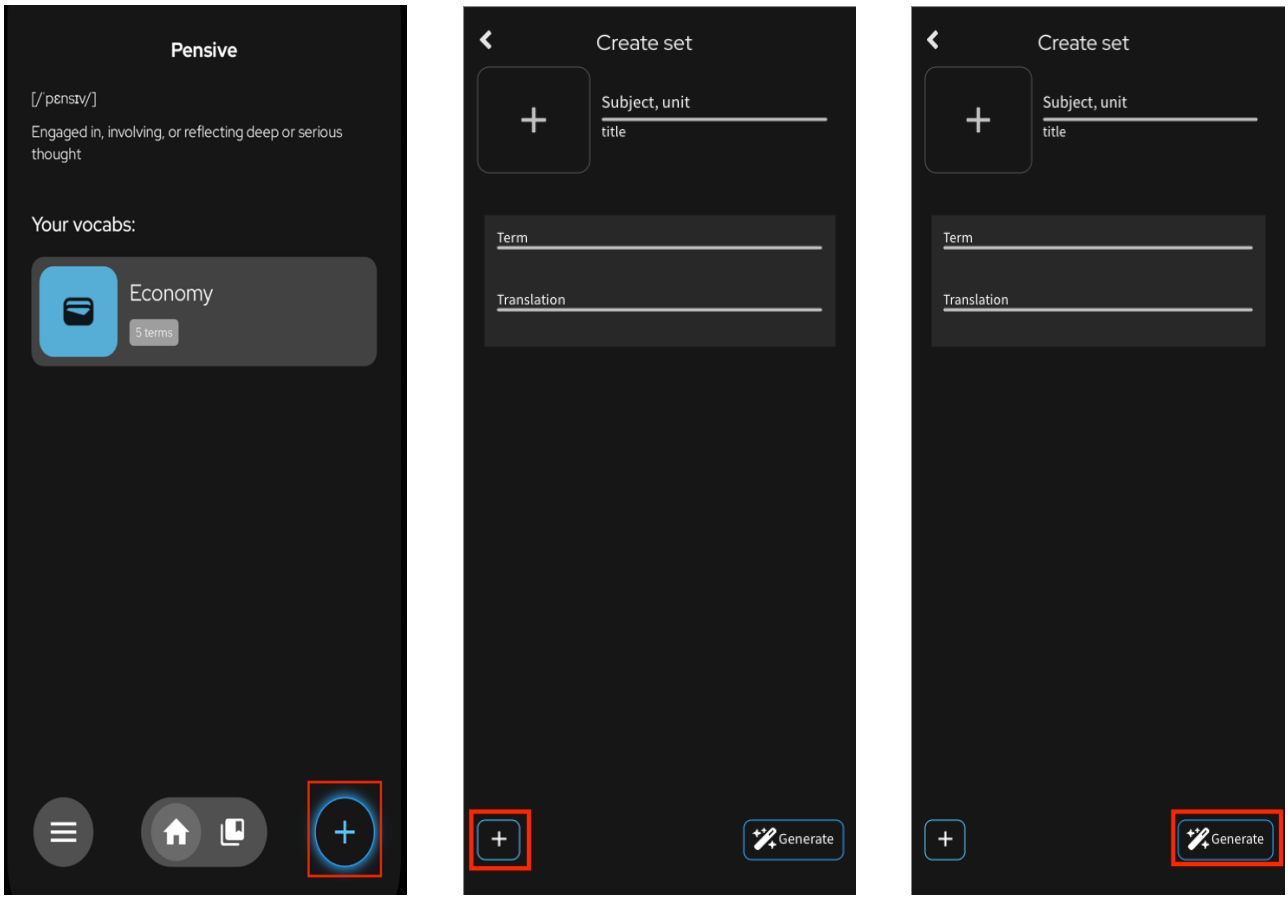

Рисунок 4.6 – Головна сторінка

На головній сторінці відображається рубрика «Слово дня», що включає в себе переклад слова, транскрипцію та пояснення. Крім того, користувач бачить кількість вивчених слів та кількість днів, коли поспіль заходив в застосунок та вивчав або практикував слова. Внизу наведені всі створені словники, які користувач ще не закінчив вивчати.

Також головна сторінка містить кнопки для навігації по застосунку, а саме: «Меню», «Головна», «Колекції» та «Створення словника».

Для того, щоб створити новий словник, необхідно перейти на сторінку «Створення словника», натиснувши на кнопку додавання в нижньому правому куті екрану (рисунок 4.7, а). На цій сторінці користувач може додати слова

власноруч, натиснувши на кнопку з плюсом (рисунок 4.7, б) або натиснути кнопку «Generate», щоб система згенерувала слово самотужки (рисунок 4.7, в).




а) б) в)

Рисунок 4.7 – Створення словника (а-в)

Після додавання усіх слів, словник потрібно зберегти задля подальшого застосування. Збереження можливе за умови, що всі поля заповнені.

Після того, як користувач успішно створив словник, відкривається сторінка з інформацією самого словника, на якій відображаються попередньо створені слова вигляді карток, методи для запам’ятовування та практикування і список слів у звичайному вигляді, щоб було зручно переглянути одразу і переклад і саме слово.

Інтерфейс сторінки словника представлено на рисунку 4.8.

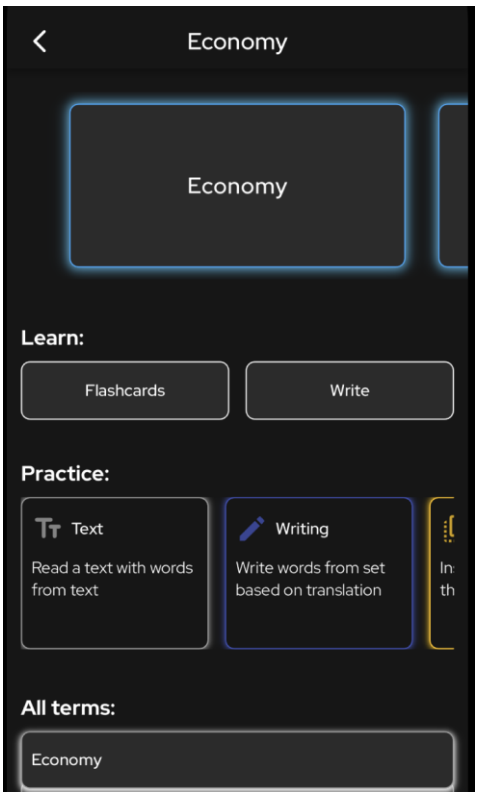



Рисунок 4.8 – Сторінка словника

Для того, щоб користувач міг ефективніше запам’ятовувати інформацію, слова подані у вигляді карток. Для того, щоб побачити переклад слово потрібно натиснути на картку з ним, після чого картка перевернеться на іншу сторону (рисунок 4.9).

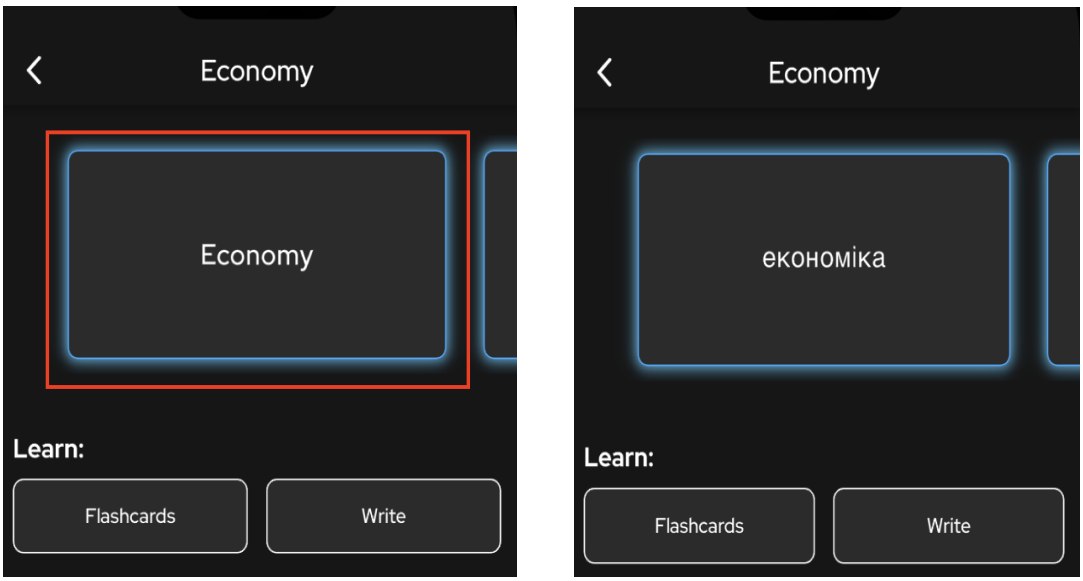
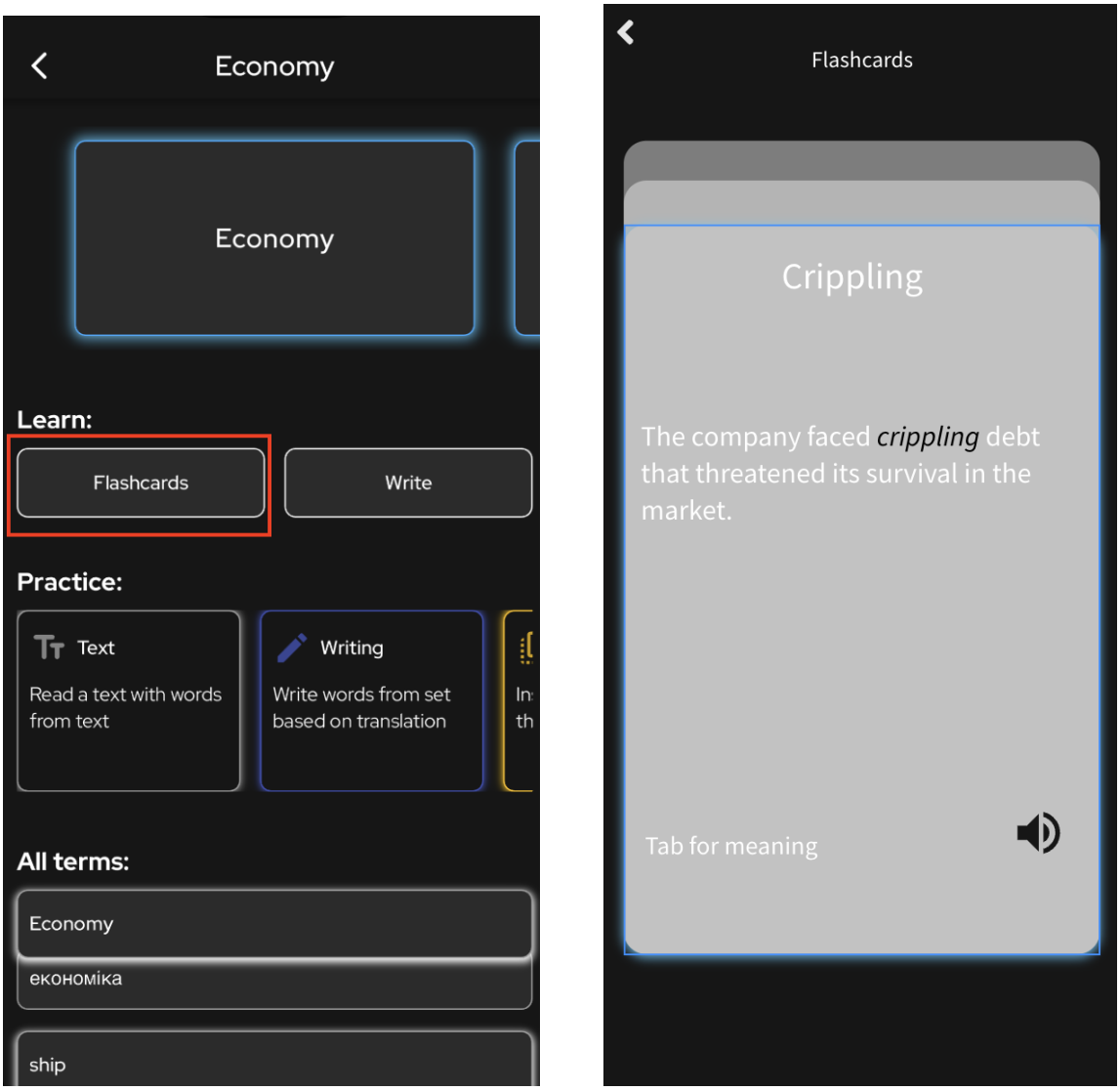



Рисунок 4.9 – Перевертання картки зі словом

На сторінці словника також є різні методи, які поліпшують процес запам’ятовування та допомагають практикувати нові слова. Одним з таким методів є «Flashcards». Цей метод призначений для прискорення та покращення процесу запам’ятовування, оскільки користувач спершу бачить слова без перекладу. Також одразу біля слова система генерує приклад з цим словом.

Для того, щоб перейти до використання методу «Flashcards» необхідно натиснути на кнопку, яка має відповідну назву (рисунок 4.10, а). Після натискання відкривається сторінка з картками (рисунок 4.10, б).




а) б)

Рисунок 4.10 – Сторінка словника (а) та сторінка «Flashcards» (б)

Другим методом для поліпшення процесу запам’ятовування є метод «Написання». Цей метод надає можливість закарбувати в пам’яті правильність написання слова, що також є важливим аспектом для подальшого застосування нових знань.

Для того, щоб відкрити сторінку з цим методом, користувач повинний натиснути на кнопку «Write» (рисунок 4.11, а). Після натискання відкривається сторінка, в середині якої розміщено переклад слова. Під цим словом є текстове поле, в яке користувач повинен написати свою відповідь. Юзер може пропустити поточне слово натискаючи на кнопку «Skip», отримати підказку натиснувши на «Hint», та відправити свій результат на перевірку «Done». Сторінка з методом «Написання» наведена на рисунку 4.11, б.

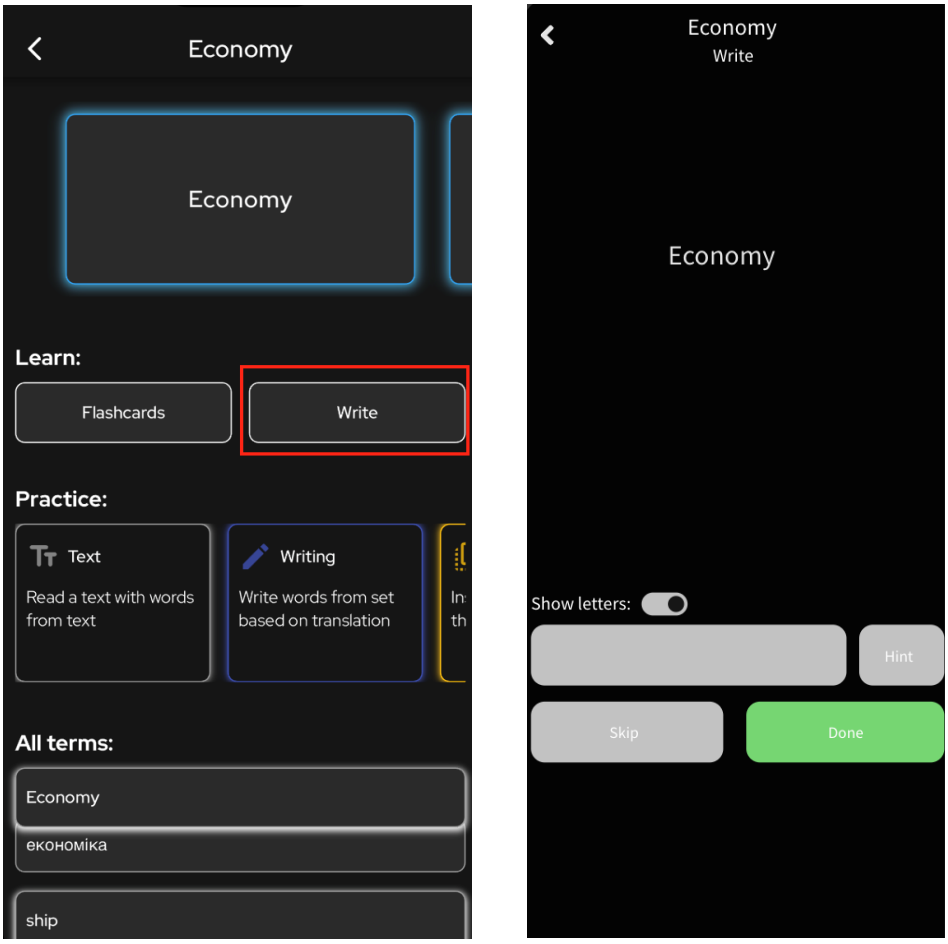
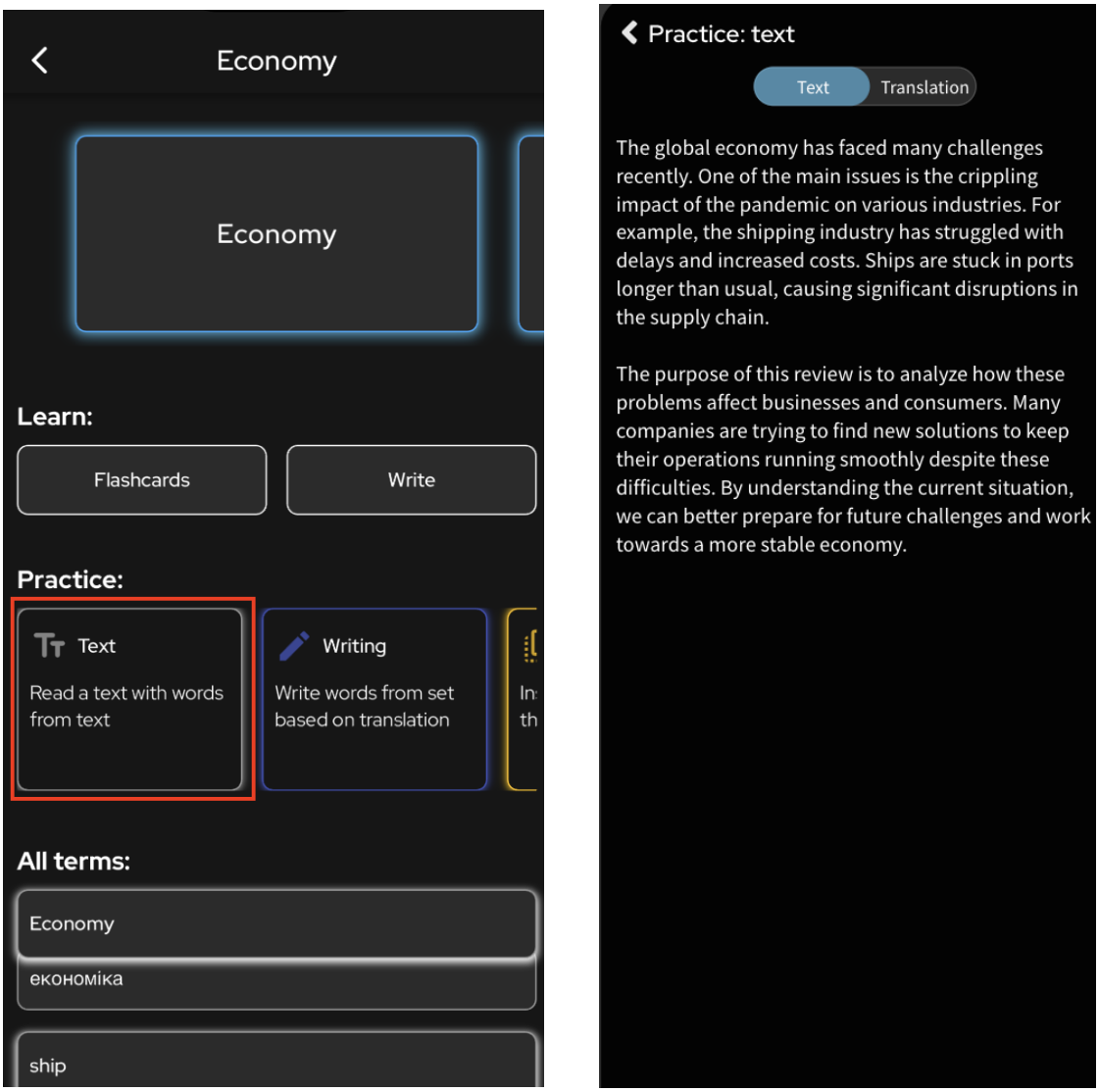



Рисунок 4.10 – Метод «Написання»

Першим методом практикування є «Текст». Основна мета цього методу полягає в наданні користувачу можливості бачити слова в контексті таким чином практикувати читання і одночасно краще засвоювати нові знання. Щоб перейти на сторінку з методом «Текст», необхідно натиснути на кнопку з текстом «Text» (рисунок 4.11, а). Після натискання відкривається сторінка з вибраним методом (рисунок 4.11, б).



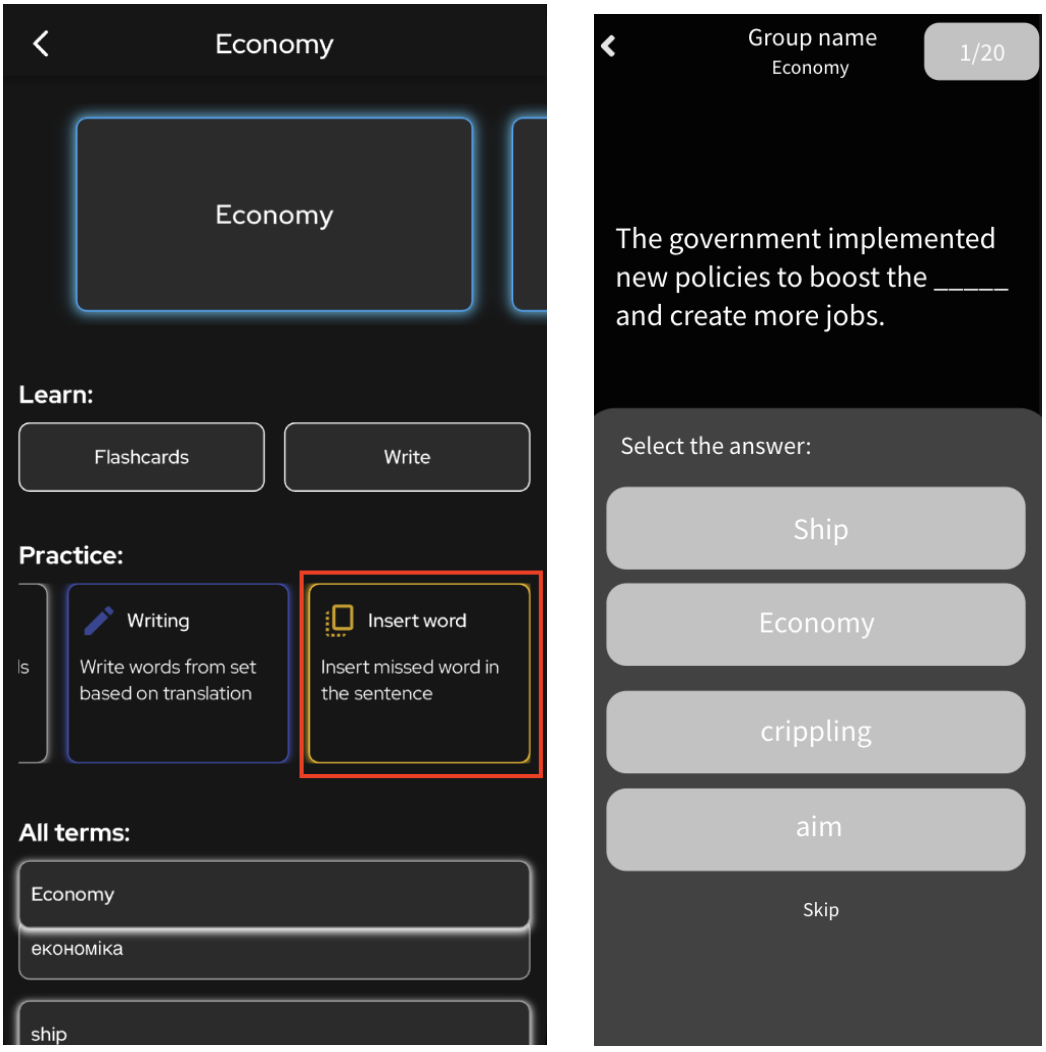

а) б)

Рисунок 4.11 – Сторінка словника (а) та відображення методу «Текст» (б)

Останнім методом практикування є «Вставка». Цей метод, як вже було описано вище, дозволяє практикувати саме застосування слова. Користувачу надається речення в якому є пропущене слово і чотири варіанти відповіді. Задача

юзера вибрати правильне слово. Якщо ж користувачу не вдалося вибрати правильну відповідь, система сповістить його повідомленням про помилку. Також юзер може пропустити завдання натискаючи на кнопку «Skip».

Для того, щоб перейти на сторінку з методом «Вставка слова», користувачу необхідно прогортати список з секції «Practice» та обрати метод «Insert word» (рисунок 4.12, а). Після натискання відкриється сторінка з методом (рисунок 4.12, б).




а) б)

Рисунок 4.12 – Сторінка словника (а) та метод «Вставка слова» (б)

Для того, щоб користувач зручно пересувався застосунком було розроблено меню. Для того, щоб його відкрити необхідно натиснути на кнопку з іконкою на головній сторінці, яка позначає меню (рисунок 4.13).

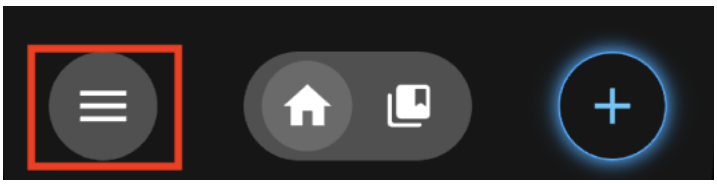


Рисунок 4.13 – Кнопка для відкриття меню

Після натискання на кнопку відкривається головне меню застосунку (рисунок 4.14), яке містить такі кнопки:

- «Home» – перехід на головну сторінку;
- «Profile» – перехід на сторінку з профілем;
- «Ukrainian (Native language)» – зміна мови, на яку відбувається переклад;
- «English (Learning)» – зміна мови, яка вивчається;
- «Sign out» – кнопка виходу з акаунту.

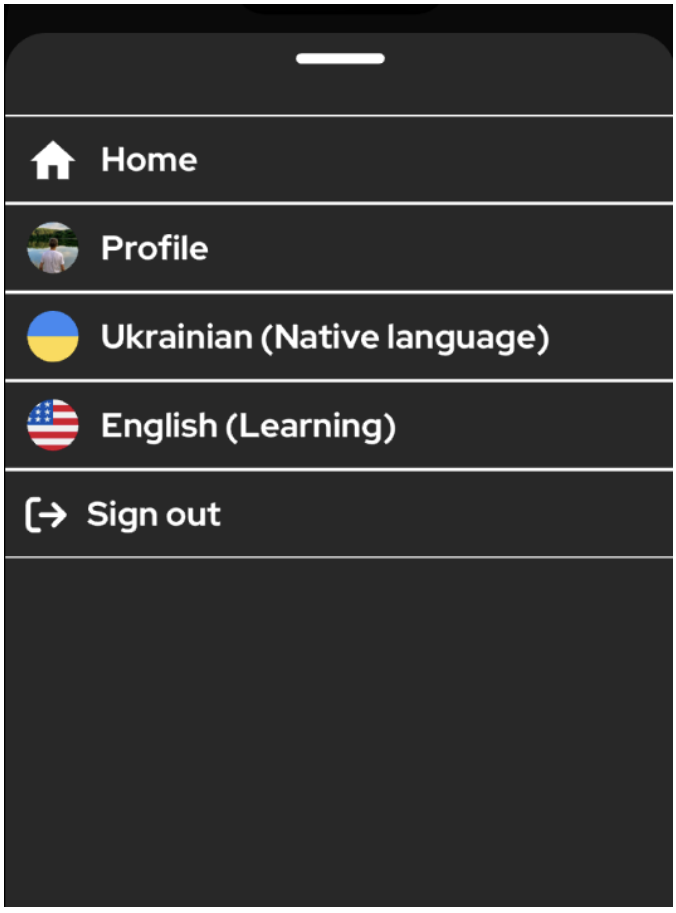



Рисунок 4.14 – Меню застосунку

Застосунок також містить сторінку «Колекція», на якій відображаються усі словники – виконані та незавершені. Для того, щоб перейти на цю сторінку потрібно на головній натиснути на відповідну кнопку з іконкою (рисунок 4.15).

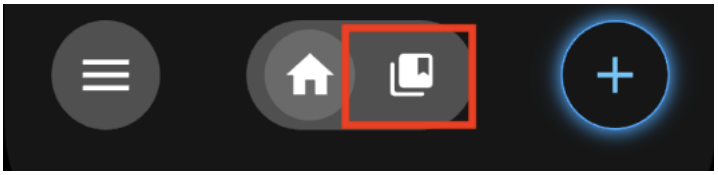


Рисунок 4.15 – Кнопка для переходу на сторінку «Колекція»

Після натискання на кнопку відкривається сторінка, на якій є всі словники користувача.

На рисунку 4.16 зображено сторінка «Колекція».


Рисунок 4.16 – Сторінка «Колекція»

Отже, було розроблено інструкцію користувача, у якій було описано навігацію застосунком, головні методи для вивчення та запам’ятовування, інструкція застосування функції для генерації слова.

### 4.4 Системні вимоги

Системними вимогами вважають такі параметри, які можуть впливати на ефективність роботи програмного забезпечення або апаратного засобу. Ці вимоги допомагають користувачам зрозуміти, чи зможе їхнє обладнання або програмне забезпечення працювати на заданих умовах.

Деталі щодо мінімальної конфігурації телефону користувача наведено в таблиці 4.1.

Таблиця 4.1 – Мінімальна конфігурація

| | Для Android | Для IOS |
|---|---|---|
| Тип процесора | 2-ядерний | Apple A8 |
| Об’єм оперативної пам’яті | 1 ГБ | 1 ГБ |
| Місце на жорсткому диску | 256 Мб | 256 Мб |
| Версія операційної системи | Android 7.0 | iOS 12 |

Деталі щодо рекомендованої конфігурації телефону наведено в таблиці 4.2.

Таблиця 4.2 – Рекомендована конфігурація

| | Для Android | Для IOS |
|---|---|---|
| Тип процесора | 4-ядерний | Apple A12 Bionic |
| Об’єм оперативної пам’яті | 2 ГБ | 2 ГБ |
| Місце на жорсткому диску | 1 ГБ | 1 ГБ |
| Версія операційної системи | Остання версія Android | Остання версія iOS |

Користувач авторизується в системі через Google або Apple акаунт. Система повідомляє про будь-яку помилку під час авторизації. Після успішної авторизації перевіряється наявність акаунту. Якщо його немає, користувач обирає мову та рівень володіння, після чого створюється акаунт. Після входу користувача на головну сторінку, система генерує слово дня та відображає статистику вивчених слів та прогрес. При переході на сторінку створення словників, система автоматично створює два пустих записи. Після натискання кнопки «Згенерувати», система створює слово відповідно до рівня та мови користувача, або користувач може заповнити поле власноруч. Після натискання кнопки збереження, система проводить валідацію всіх полів. Якщо є помилка, система відображає її та вимагає заповнити всі поля. У випадку успішної валідації, словник зберігається в базі даних.

Таким чином, інструкція користувача є ключовим документом, який сприяє успішному використанню продукту чи послуги, забезпечуючи користувачам необхідну інформацію та вказівки для ефективного взаємодії з ним. Вона допомагає уникнути непорозумінь, помилок та зберігає час користувача, надаючи чітку та структуровану інформацію.

### 4.4 Висновки

У четвертому розділі проведено детальне тестування функціональних та графічних компонентів мобільного застосунку з метою перевірки їх працездатності. Враховуючи результати тестування було розроблено керівництво користувача для системи, а також встановлено системні вимоги, що містять необхідну інформацію щодо конфігурації пристрою користувача з операційною системою Android або IOS.

# ВИСНОВКИ

У бакалаврській дипломній роботі було розроблено програмний мобільний застосунок для розширення словникового запасу, який сприяє пришвидшенню процесу пошуку нових слів для вивчення та їх ефективному запам’ятовуванню.

Було проаналізовано стан способів розширення словникового запасу та напрямків удосконалення програмного забезпечення для розширення словникового запасу, виконано порівняльний аналіз аналогів, поставлено задачі та розглянуто впровадження штучного інтелекту в навчанні.

Було удосконалено метод знаходження слів для вивчення за допомогою використання штучного інтелекту, який дозволяє підбирати нові слова відносно мови для вивчення та рівня користувача, що дозволило створювати індивідуальні набори під кожного користувача та їх потреби.

Було удосконалено методи практикування та запам’ятовування нових слів на основі штучного інтелекту, а саме: «Флеш-карти», «Текст», «Вставка слова». Удосконалені методи підвищують ефективність в навчанні, оскільки інтерактивні та контекстуальні підходи роблять процес навчання більш захоплюючим і результативний.

Було розроблено модель застосунку, алгоритм авторизації та створення профілю, а також алгоритм створення словника з генерацією слів.

Крім того, було проведено варіантний аналіз і обґрунтувано вибір засобів для реалізації програмного забезпечення. Розроблено модуль для створення і генерації словника, сервіс для реєстрації користувача та методи для запам’ятовування та швидкого засвоєння нових слів. Для розробки було використано мову програмування Dart, фреймворк Flutter в поєднанні хмарним сервісом Firebase від Google, середовище програмної розробки Visual Studio Code.

Було розглянуто рівні тестування програмного забезпечення, а тестування програмного застосунку показало, що проблем не було виявлено.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ткачик О.В., Роговська Н.В. Шляхи поповнення лексичного складу сучасної англійської мови. Вісник. 2017. № 9. С. 36-40.

2. Подосиннікова Г. І. Методична характеристика інтерактивного навчання іноземних мов / Г. І. Подосиннікова. - К.: ТОВ «Видавниче підприємство «Едельвейс», 2012. – 208 с.

3. Інтерактивні методи навчання: переваги та використання в сучасній освіті [Електронний ресурс] – Режим доступу до ресурсу: https://gosta.media/nauka-ta- osvita/interaktyvni-metody-navchannia-perevahy-ta-vykorystannia-v-suchasnij-osviti/ (дата звернення: 08.05.2024).

4. Степанчук П.В. Використання хмарних сервісів у мобільній розробці для підвищення захисту, продуктивності та функціональності / П.В. Степанчук, О.В. Романюк // Матеріали LIII Науково-технічної конференції підрозділів Вінницького національного технічного університету, Вінниця, 2024. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/20673

5. Степанчук П.В. Використання штучного інтелекту у мобільній розробці для підбору слів для вивчення / П.В. Степанчук, О.В. Романюк // Молодь в науці: дослідження, проблеми, перспективи, Вінниця, 2024. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/21683

6. Активний та пасивний словниковий запас [Електронний ресурс] – Режим доступу до ресурсу: https://www.englishdom.com/ua/blog/shho-take-pasivnij-i- aktivnij-slovnikovij-zapas/ (дата звернення 10.05.2024).

7. Методи засвоєння інформації учнями [Електронний ресурс] – Режим доступу до ресурсу: https://www.04141.com.ua/list/402501 (дата звернення 10.05.2024).

8. Quizlet [Електронний ресурс] – Режим доступу до ресурсу: https://quizlet.com/ua (дата звернення: 10.05.2024).

9. Babbel [Електронний ресурс] – Режим доступу до ресурсу: https://ua.babbel.com/ (дата звернення: 10.05.2024).

10. Anki [Електронний ресурс] – Режим доступу до ресурсу: https://www.ankiapp.com/ (дата звернення: 10.05.2024).

11. Using Flashcards [Електронний ресурс] – режим доступу до ресурсу: https://usm.maine.edu/learning-commons/using-flash-cards/ (дата звернення 10.05.2024).

12. Mobile Operating System Market Share Worldwide [Електронний ресурс] – Режим доступу до ресурсу: https://gs.statcounter.com/os-market-share/mobile/worldwide (дата звернення 10.05.2024).

13. iPhone vs Android Statistics [Електронний ресурс] – Режим доступу до ресурсу: https://backlinko.com/iphone-vs-android-statistics (дата звернення 10.05.2024).

14. 10 Ways AI in Education is Transforming the Industry [Електронний ресурс] – Режим доступу до ресурсу: https://appinventiv.com/blog/10-ways-artificial-intelligence-transforming-the-education-industry/ (дата звернення 10.05.2024).

15. What Is NoSQL database [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/topics/nosql-databases (дата звернення 12.05.2024).

16. Dart programming language [Електронний ресурс] – Режим доступу до ресурсу: https://dart.dev/ (дата звернення 12.05.2024).

17. Deven Joshi. Building Cross-Platform Apps with Flutter and Dart. Delhi : BPB Publications, 2023. 378 p.

18. Open GL ES Overview [Електронний ресурс] – Режим доступу до ресурсу: https://www.khronos.org/opengles/ (дата звернення 12.05.2024).

19. JetBrains [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/ (дата звернення 12.05.2024).

20. Frahaan Hussain. Building Mobile Magic: Integrating Flutter with Firebase. London : Sonar Publishing, 2024. 412 p.

21. BLoC [Електронний ресурс] – Режим доступу до ресурсу: https://bloclibrary.dev (дата звернення: 20.03.2024).

22. Переваги дизайну UI [Електронний ресурс] – Режим доступу до ресурсу: https://www.mindseo.com/resource/user-interface-design/ (дата звернення: 20.03.2024).

23. Інтеграційне тестування [Електронний ресурс] – Режим доступу до ресурсу: https://testsigma.com/guides/integration-testing/ (дата звернення: 25.03.2024).

24. Золоте тестування на Flutter [Електронний ресурс] – Режим доступу до ресурсу: https://tech.appunite.com/blog/golden-testing-with-flutter?redirected=true (дата звернення: 25.03.2024).

25. Мокінг і ін’єкція залежностей [Електронний ресурс] – Режим доступу до ресурсу: https://quickcoder.org/flutter-dependency-mocking/ (дата звернення: 25.03.2024).

# ДОДАТКИ

69

# ДОДАТОК А

## Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. каф. ПЗ

_____ О. Н. Романюк

« 12 » березня 2024 р.

**Технічне завдання**
**на бакалаврську дипломну роботу**
**«Розробка програмного мобільного застосунку для розширення словникового запасу»**
**за спеціальністю 121 – Інженерія програмного забезпечення**

Керівник бакалаврської дипломної роботи:

_____ к.т.н., доцент каф. ПЗ Романюк О.В.

« 12 » березня 2024 р.

Виконав:

_____ Студент гр. ЗПІ-20б Степанчук П.В.

« 12 » березня 2024 р.

Вінниця – 2024 року

**1. Найменування та галузь застосування.**

Бакалаврська дипломна робота: «Розробка програмного мобільного застосунку для розширення словникового запасу».

Галузь застосування – мобільні технології.

**2. Підстава для розробки.**

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ №80 від «11» березня 2024 р. ректора по ВНТУ про закріплення тем БДР.

**3. Мета та призначення розробки.**

Метою роботи є удосконалення процесу розширення словникового запасу за допомогою розробки мобільного застосунку, що сприятиме пришвидшенню процесу пошуку нових слів для вивчення та ефективному їх запам’ятовуванню.

Призначення роботи – розробка та реалізація програмного мобільного застосунку для розширення словникового запасу.

**4. Вихідні дані для проведення НДР.**

Перелік основних літературних джерел, на основі яких буде виконуватись БДР.

1. Deven Joshi. Building Cross-Platform Apps with Flutter and Dart. Delhi : BPB Publications, 2023. 378 p.

2. Frahaan Hussain. Building Mobile Magic: Integrating Flutter with Firebase. London : Sonar Publishing, 2024. 412 p.

3. Five tips to improve your English vocabulary [Електронний ресурс] – Режим доступу до ресурсу: https://learnenglish.britishcouncil.org/english-levels/improve-your-english-level/five-tips-improve-your-english-vocabulary

4. Why Is Vocabulary Expansion Necessary? [Електронний ресурс] – Режим доступу до ресурсу: https://spellquiz.com/blog/vocabulary-expansion

**5. Технічні вимоги.**

Тип програми – програмний мобільний застосунок; модель розробки – ітеративна; засоби організації даних програми – фреймворки Flutter та Firebase; вхідні дані: мова для вивчення та рівень володіння нею; вихідні дані: створені словники; середовище розробки – Visual Studio Code; мова програмування – Dart; програмне забезпечення для симуляції мобільних пристроїв – Xcode.

**6. Конструктивні вимоги.**

Графічна та текстова документація повинна відповідати діючим стандартам України.

**7. Перелік технічної документації, що пред'являється по закінченню робіт:**

– пояснювальна записка до БДР;
– технічне завдання;
– лістинги програми.

**8. Вимоги до рівня уніфікації та стандартизації**

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

**9. Стадії та етапи розробки:**

| Ч.ч. | Назва етапів бакалаврської дипломної роботи | Строк виконання етапів роботи |
|---|---|---|
| 1 | Аналіз стану питання та постановка задач дослідження | 14.03.24 – 21.03.2024 |
| 2 | Розробка моделі застосунку | 22.03.2024 – 25.03.2024 |
| 3 | Удосконалення методу знаходження слів для вивчення за допомогою використання штучного інтелекту | 27.03.2024 – 01.04.2024 |
| 4 | Удосконалення методів практикування та запам’ятовування нових слів на основі штучного інтелекту | 02.04.2024 – 07.04.2024 |
| 5 | Розробка алгоритму створення словника | 08.04.2024 – 14.04.2024 |
| 6 | Аналіз та вибір засобів для реалізації програмного продукту | 15.04.2024 – 18.04.2024 |
| 7 | Розробка програмного забезпечення | 19.04.2024 – 09.05.2024 |
| 8 | Тестування програмного застосунку | 10.05.2024 – 23.05.2024 |
| 9 | Оформлення матеріалів до захисту БДР | 24.05.2024 – 30.05.24 |

**10. Порядок контролю та прийняття.**

Виконання етапів бакалаврської дипломної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття бакалаврської дипломної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

73

## ДОДАТОК Б

### Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Розробка програмного мобільного застосунку для розширення словникового запасу»

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Романюк О.В.

| | |
|---|---|
| Оригінальність | 95,2 % |
| Схожість | 4,8 % |

**Аналіз звіту подібності**

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Степанчук П.В.

Керівник роботи _____ Романюк О.В.

# ДОДАТОК В

## Лістинг програми

```
import 'package:flutter/material.dart';

import 'package:jelly_vocab/localization/index.dart';
import 'app.dart';
import 'app_initialization.dart';

void main() async {
  await initializeApp();

  runApp(
    EasyLocalization(
      path: CodegenLoader.path,
      supportedLocales: CodegenLoader.supportedLocales,
      fallbackLocale: CodegenLoader.supportedLocales.last,
      assetLoader: const CodegenLoader(),
      child: JellyVocab(),
    ),
  );
}

import 'package:flutter/material.dart';

import 'package:easy_localization/easy_localization.dart';

import 'package:jelly_vocab/app_state_wrapper.dart';
import 'package:jelly_vocab/router/index.dart';
import 'package:jelly_vocab/services/index.dart';
import 'package:jelly_vocab/styles/index.dart';

class JellyVocab extends StatelessWidget {
  JellyVocab({super.key});

  final _appRouter = getIt<AppRouter>();

  @override
  Widget build(BuildContext context) {
    return AppStateWrapper(
      child: MaterialApp.router(
        debugShowCheckedModeBanner: false,
        theme: AppTheme.getAppTheme(context),
        routerConfig: _appRouter.config(
          navigatorObservers: () => [RouterObserver(LoggerService.instance)],
        ),
        localizationsDelegates: context.localizationDelegates,
```

```
        supportedLocales: context.supportedLocales,
        locale: context.locale,
      ),
    );
  }
}

import 'package:flutter/material.dart';

import 'package:flutter_bloc/flutter_bloc.dart';

import 'package:jelly_vocab/blocs/index.dart';
import 'package:jelly_vocab/services/index.dart';

class AppStateWrapper extends StatelessWidget {
  const AppStateWrapper({
    super.key,
    required this.child,
  });

  final Widget child;

  @override
  Widget build(BuildContext context) {
    return MultiBlocProvider(
      providers: [
        BlocProvider(
          create: (context) => getIt<AuthBloc>(),
        ),
      ],
      child: child,
    );
  }
}

import 'package:flutter/material.dart';

import 'package:easy_localization/easy_localization.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_dotenv/flutter_dotenv.dart';
import 'package:flutter_native_splash/flutter_native_splash.dart';
import 'package:hive/hive.dart';
import 'package:path_provider/path_provider.dart';

import 'package:jelly_vocab/blocs/observer.dart';
import 'package:jelly_vocab/firebase_options.dart';
import 'package:jelly_vocab/models/index.dart';
```

```
import 'package:jelly_vocab/services/index.dart';

Future<void> initializeApp() async {
  final widgetsBinding = WidgetsFlutterBinding.ensureInitialized();

  initializeSplashScreen(widgetsBinding);
  await initializeDotenv();
  await initializeLogger();
  await initializeLocalization();
  await initializeFirebase();
  await initHive();
  initializeCrashlytics();
  initializeBlocObserver();
  configureAuthDependencies();
}

Future<void> initializeFirebase() {
  return Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
}

void initializeSplashScreen(WidgetsBinding widgetsBinding) {
  FlutterNativeSplash.preserve(widgetsBinding: widgetsBinding);
}

Future<void> initHive() async {
  final applicationPath = (await getApplicationDocumentsDirectory()).path;

  Hive
    ..init(applicationPath)
    ..registerAdapter(WordOfTheDayImplAdapter());
}

Future<void> initializeLogger() {
  return LoggerService.instance.init();
}

Future<void> initializeLocalization() {
  return EasyLocalization.ensureInitialized();
}

Future<void> initializeDotenv() {
  return dotenv.load();
}

void initializeCrashlytics() {
  // FlutterError.onError = (errorDetails) {
```

```
  //   LoggerService.instance.logError(errorDetails.exception, errorDetails.stack);
  // };

  // PlatformDispatcher.instance.onError = (error, stack) {
  //   LoggerService.instance.logError(error, stack);

  //   return true;
  // };
}

void initializeBlocObserver() {
  Bloc.observer = SimpleBlocObserver(LoggerService.instance);
}
import 'package:flutter/material.dart';

import 'package:jelly_vocab/models/vocabulary/models.dart';
import 'package:jelly_vocab/styles/index.dart';

class VocabularyCard extends StatelessWidget {
  const VocabularyCard({
    super.key,
    required this.item,
    required this.onTap,
  });

  final VocabularyModel item;

  final VoidCallback onTap;

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: onTap,
      child: Container(
        padding: const EdgeInsets.all(8),
        decoration: BoxDecoration(
          color: AppColors.greyShadow,
          borderRadius: BorderRadius.circular(12),
        ),
        child: Row(
          children: [
            _VocabularyAvatar(
              vocabularyModel: item,
            ),
            const SizedBox(width: 12),
            Expanded(
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
```

```
              children: [
                Text(
                  item.name,
                  style: const TextStyle(
                    fontSize: 20,
                    color: Colors.white,
                  ),
                ),
                const SizedBox(height: 10),
                Container(
                  padding: const EdgeInsets.symmetric(
                    vertical: 4,
                    horizontal: 6,
                  ),
                  decoration: BoxDecoration(
                    color: Colors.grey,
                    borderRadius: BorderRadius.circular(4),
                  ),
                  child: Text(
                    '${item.words.length} terms',
                    style: const TextStyle(
                      fontSize: 11,
                      color: Colors.white,
                    ),
                  ),
                ),
              ],
            ),
          ),
        ],
      ),
    ),
  );
 }
}

class _VocabularyAvatar extends StatelessWidget {
 const _VocabularyAvatar({
  required this.vocabularyModel,
 });

 final VocabularyModel vocabularyModel;

 @override
 Widget build(BuildContext context) {
  return Container(
   width: 78,
   height: 78,
```

```
      alignment: Alignment.center,
      decoration: BoxDecoration(
        color: AppColors.primary.withOpacity(0.8),
        borderRadius: BorderRadius.circular(14),
      ),
      child: const Icon(
        Icons.wallet,
        size: 36,
      ),
    );
  }
}

import 'dart:async';

import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:freezed_annotation/freezed_annotation.dart';
import 'package:google_sign_in/google_sign_in.dart';
import 'package:injectable/injectable.dart';

import 'package:jelly_vocab/models/index.dart';
import 'package:jelly_vocab/repositories/index.dart';

part 'auth_bloc.freezed.dart';
part 'auth_event.dart';
part 'auth_state.dart';

@Singleton(scope: 'auth')
class AuthBloc extends Bloc<AuthEvent, AuthState> {
  final AuthRepository authRepository;
  final UserRepository userRepository;

  late StreamSubscription<User?> _subscription;

  AuthBloc({
    required this.authRepository,
    required this.userRepository,
  }) : super(const AuthState()) {
    _subscription = authRepository.authenticationStatus.listen((user) {
      add(AuthEvent.authenticationStateChanged(user));
    });

    on<_AuthenticationStateChanged>(_authenticationStatusChanged);

    on<_SignInWithGoogle>(_signInWithGoogle);
    on<_SignInWithApple>(_signInWithApple);
    on<_FinishOnboarding>(_finishOnboarding);
```

```
  on<_SignOut>(_signOut);
}

// TODO(Pasha): Refactor in FormBloc
void signInWithGoogle() => add(const _SignInWithGoogle());
void singInWithApple() => add(const _SignInWithApple());

void finishOnboarding(UserProfile userProfile) =>
    add(_FinishOnboarding(userProfile));

FutureOr<void> _authenticationStatusChanged(
  _AuthenticationStateChanged event,
  Emitter<AuthState> emit,
) async {
  final firebaseUser = event.user;

  if (firebaseUser == null) {
    return emit(AuthState.unauthorized());
  }

  final userProfile = await userRepository.getUserProfile(
    firebaseUser.uid,
  );

  if (userProfile != null) {
    return emit(
      AuthState.authorized(firebaseUser, userProfile),
    );
  }

  emit(
    state.copyWith(
      firebaseUser: firebaseUser,
      status: AuthorizationStatus.onboarding,
    ),
  );
}

// TODO(Pasha): Refactor into formBloc
FutureOr<void> _signInWithGoogle(
  _SignInWithGoogle event,
  Emitter<AuthState> emit,
) async {
  try {
    final googleUser = await GoogleSignIn().signIn();

    final googleAuth = await googleUser?.authentication;
```

```
      final credential = GoogleAuthProvider.credential(
        accessToken: googleAuth?.accessToken,
        idToken: googleAuth?.idToken,
      );

      await authRepository.signInWithGoogle(credential);
    } catch (e, stackTrace) {
      addError(e, stackTrace);

      // TODO(Pasha): add error handling
    }
  }

  // TODO(Pasha): Refactor into formBloc
  FutureOr<void> _signInWithApple(
    _SignInWithApple event,
    Emitter<AuthState> emit,
  ) async {
    // TODO(Pasha): Add apple sign in
  }

  FutureOr<void> _finishOnboarding(
    _FinishOnboarding event,
    Emitter<AuthState> emit,
  ) async {
    emit(
      state.copyWith(
        user: event.userProfile,
        status: AuthorizationStatus.authorized,
      ),
    );
  }

  FutureOr<void> _signOut(_SignOut event, Emitter<AuthState> emit) {
    return authRepository.signOut();
  }

  @override
  Future<void> close() async {
    await _subscription.cancel();
    return super.close();
  }
}

part of 'auth_bloc.dart';

@freezed
```

```
class AuthEvent with _$AuthEvent {
  const factory AuthEvent.authenticationStateChanged(
    User? user,
  ) = _AuthenticationStateChanged;

  const factory AuthEvent.signInWithGoogle() = _SignInWithGoogle;
  const factory AuthEvent.signInWithApple() = _SignInWithApple;
  const factory AuthEvent.finishOnboarding(UserProfile userProfile) =
      _FinishOnboarding;

  const factory AuthEvent.signOut() = _SignOut;
}

part of 'auth_bloc.dart';

enum AuthorizationStatus {
  initial,
  onboarding,
  authorized,
  unauthorized,
}

@freezed
class AuthState with _$AuthState {
  const AuthState._();

  const factory AuthState({
    UserProfile? user,
    User? firebaseUser,
    @Default(AuthorizationStatus.initial) AuthorizationStatus status,
  }) = _AuthState;

  bool get isAuth => user != null;

  factory AuthState.authorized(
    User? firebaseUser,
    UserProfile user,
  ) {
    return AuthState(
      user: user,
      firebaseUser: firebaseUser,
      status: AuthorizationStatus.authorized,
    );
  }

  factory AuthState.unauthorized() {
    return const AuthState(
      status: AuthorizationStatus.unauthorized,
```

```
    );
  }
}

import 'dart:async';

import 'package:darq/darq.dart';
import 'package:hive/hive.dart';
import 'package:injectable/injectable.dart';

import 'package:jelly_vocab/blocs/index.dart';
import 'package:jelly_vocab/core/index.dart';
import 'package:jelly_vocab/models/index.dart';
import 'package:jelly_vocab/repositories/words_repository.dart';

typedef WordOfTheDayState = NetworkListState<WordOfTheDay>;

@lazySingleton
class WordOfTheDayBloc
    extends NetworkListBloc<WordOfTheDay, WordOfTheDayState> {
  WordOfTheDayBloc({
    required this.authBloc,
    required this.repository,
  }) : super(
          const WordOfTheDayState(
            data: [],
          ),
        );

  Box<WordOfTheDay>? box;

  final AuthBloc authBloc;
  final WordsRepository repository;

  @override
  Future<List<WordOfTheDay>> onLoadAsync() async {
    final now = DateTime.now();
    box ??= await Hive.openBox(EnvConstants.wordOfTheDayBoxName);

    final words = box!.values.toList();

    var todaysWord = words.firstWhereOrDefault(
      (word) => word.date?.isSameDate(now) ?? false,
    );

    if (todaysWord == null) {
      final userProfile = authBloc.state.user!;
```

```
      todaysWord = await repository.createWordOfTheDay(
        userProfile.studyLevel,
        userProfile.language,
      );

      if (todaysWord != null) {
        words.add(todaysWord);
        unawaited(box!.add(todaysWord));
      }
    }

    return words.reversed.toList();
  }

  @override
  bool equals(WordOfTheDay item1, WordOfTheDay item2) {
    return item1.name == item2.name;
  }
}

import 'package:firebase_auth/firebase_auth.dart';
import 'package:injectable/injectable.dart';

@Injectable(scope: 'auth')
class AuthRepository {
  final client = FirebaseAuth.instance;

  Stream<User?> get authenticationStatus => client.authStateChanges();

  Future<UserCredential> signInWithGoogle(
    OAuthCredential credential,
  ) async {
    return client.signInWithCredential(credential);
  }

  Future<void> signOut() {
    return Future.value();
  }
}

import 'package:cloud_firestore/cloud_firestore.dart';

abstract class FirestoreRepoBase<T> {
  final String userUid;

  final client = FirebaseFirestore.instance;

  FirestoreRepoBase({required this.userUid});
```

```
  CollectionReference<T> collectionReference();

  DocumentReference userDoc() {
    return client.collection('users').doc(userUid);
  }
}

import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:injectable/injectable.dart';

import 'package:jelly_vocab/models/vocabulary/models.dart';
import 'firestore_repo_base.dart';

@injectable
class VocabularyRepository extends FirestoreRepoBase<VocabularyModel> {
  VocabularyRepository({
    @Named('userUid') required super.userUid,
  });

  @override
  CollectionReference<VocabularyModel> collectionReference() {
    return userDoc().collection('vocabularies').withConverter(
          fromFirestore: (snapshot, _) =>
              VocabularyModel.fromSnapshot(snapshot),
          toFirestore: (payload, _) => payload.toJson(),
        );
  }

  Future<List<VocabularyModel>> getVocabularies() async {
    final response = await collectionReference().get();

    return response.docs.map((doc) => doc.data()).toList();
  }

  Future<List<VocabularyModel>> getUncompletedVocabularies() async {
    final response =
        await collectionReference().where('completed', isEqualTo: false).get();

    return response.docs.map((doc) => doc.data()).toList();
  }

  Future<VocabularyModel> createVocabulary(VocabularyModel payload) async {
    final response = await collectionReference().add(payload);

    return payload.copyWith(
      uid: response.id,
    );
```

```
  }
}

import 'package:injectable/injectable.dart';

import 'package:jelly_vocab/models/index.dart';
import 'package:jelly_vocab/models/vocabulary/models.dart';
import 'package:jelly_vocab/repositories/index.dart';

typedef HomeState = NetworkListState<VocabularyModel>;

@lazySingleton
class HomeBloc extends NetworkListBloc<VocabularyModel, HomeState> {
  final VocabularyRepository repository;

  HomeBloc({
    required this.repository,
  }) : super(const HomeState(data: []));

  @override
  Future<List<VocabularyModel>> onLoadAsync() {
    return repository.getUncompletedVocabularies();
  }

  @override
  bool equals(VocabularyModel item1, VocabularyModel item2) {
    return item1.uid == item2.uid;
  }
}

import 'package:flutter/material.dart';

import 'package:flutter_bloc/flutter_bloc.dart';

import 'package:jelly_vocab/router/index.dart';
import 'package:jelly_vocab/styles/index.dart';
import 'package:jelly_vocab/widgets/vocabulary_card.dart';
import 'home_bloc.dart';
import 'widgets/words_of_the_day_list.dart';

// TODO(Pasha): Refactor to sliver
@RoutePage()
class HomeScreen extends StatelessWidget {
  const HomeScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

```
body: SafeArea(
  bottom: false,
  child: Padding(
    padding: const EdgeInsets.symmetric(horizontal: 24),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        const SizedBox(height: 20),
        const WordsOfTheDayList(),
        const Text(
          'Your vocabs:',
          style: TextStyle(
            fontSize: 18,
            color: AppColors.white,
            fontWeight: FontWeight.w500,
          ),
        ),
        BlocBuilder<HomeBloc, HomeState>(
          builder: (_, state) {
            final vocabularies = state.data;

            return Expanded(
              child: ListView.separated(
                itemCount: vocabularies.length,
                padding: const EdgeInsets.only(
                  top: 16,
                  bottom: 30,
                ),
                separatorBuilder: (_, index) {
                  return const SizedBox(height: 10);
                },
                itemBuilder: (_, index) {
                  final vocabulary = vocabularies[index];

                  return VocabularyCard(
                    item: vocabulary,
                    onTap: () => context.pushRoute(
                      VocabularyRoute(
                        vocabulary: vocabulary,
                      ),
                    ),
                  );
                },
              ),
            );
          },
        ),
      ],
```

# ДОДАТОК Г

## Ілюстративна частина

## ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ РОЗШИРЕННЯ СЛОВНИКОВОГО ЗАПАСУ

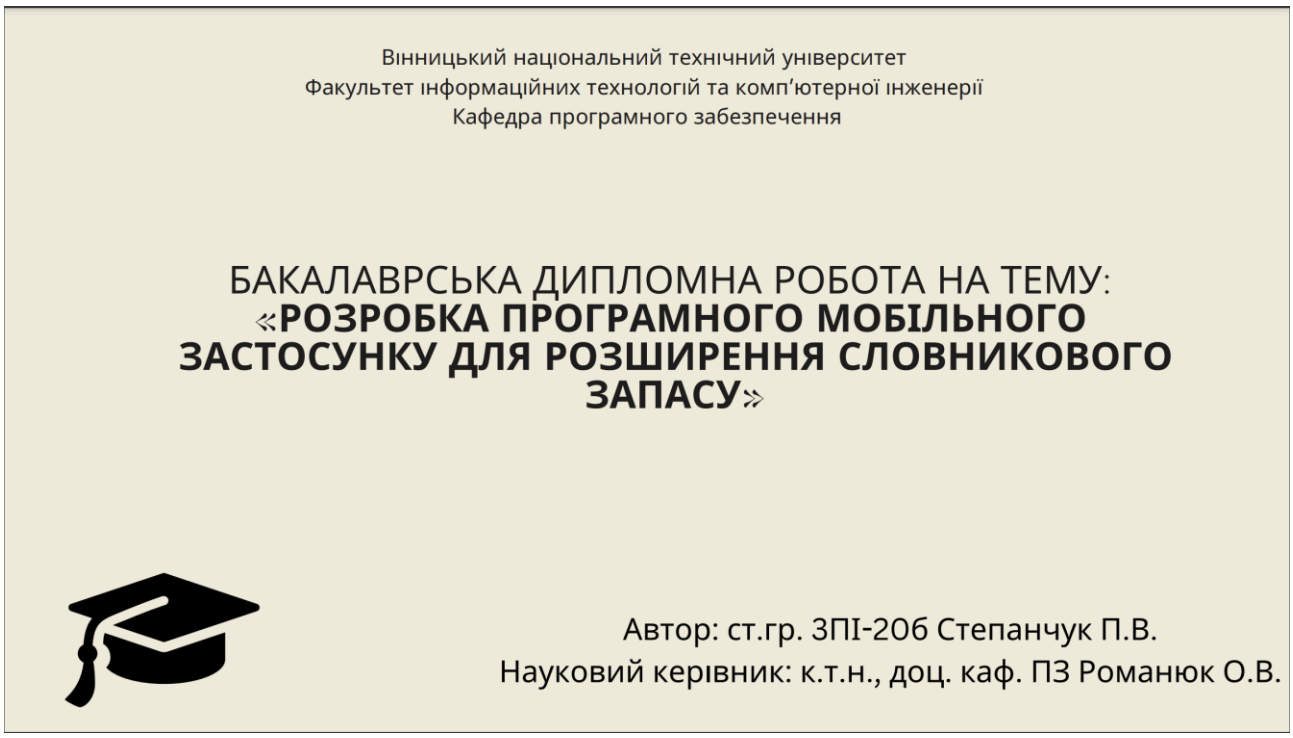



Рисунок Г.1 – Титульний слайд

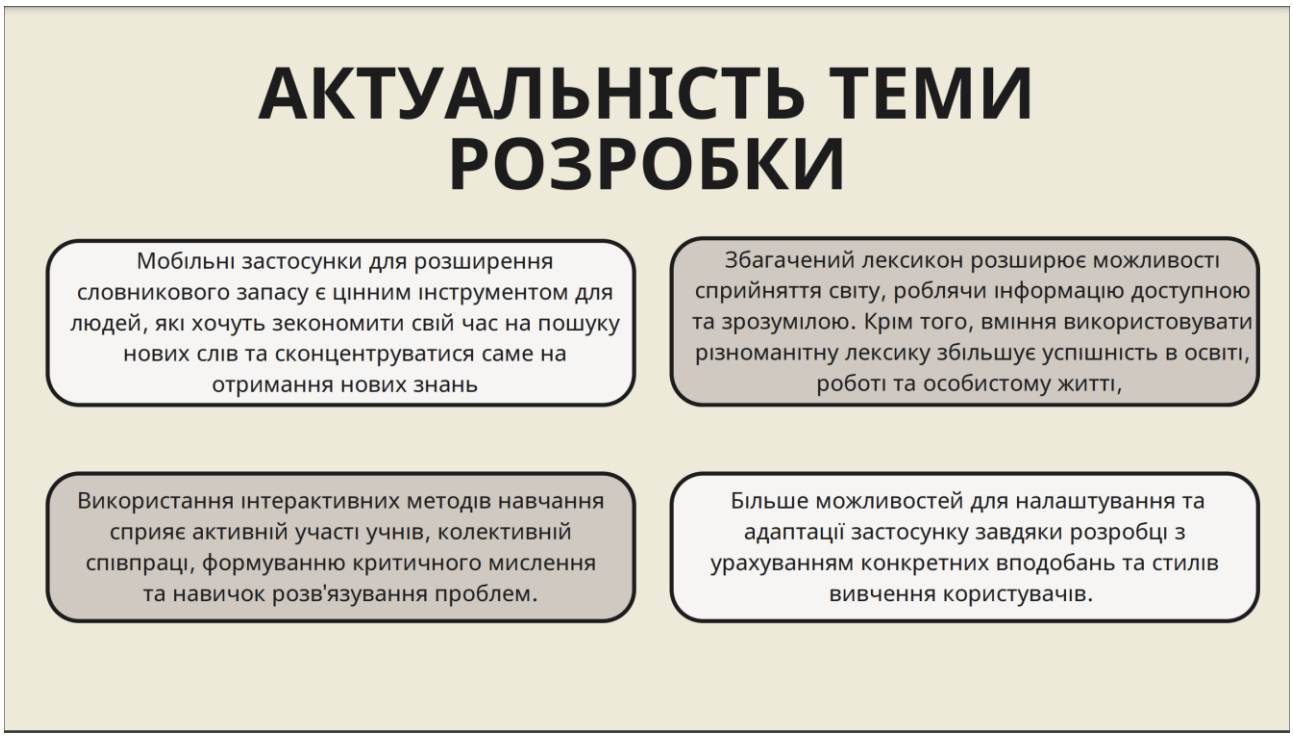



Рисунок Г.2 – Актуальність теми розробки


Рисунок Г.3 – Мета, об’єкт та предмет дослідження

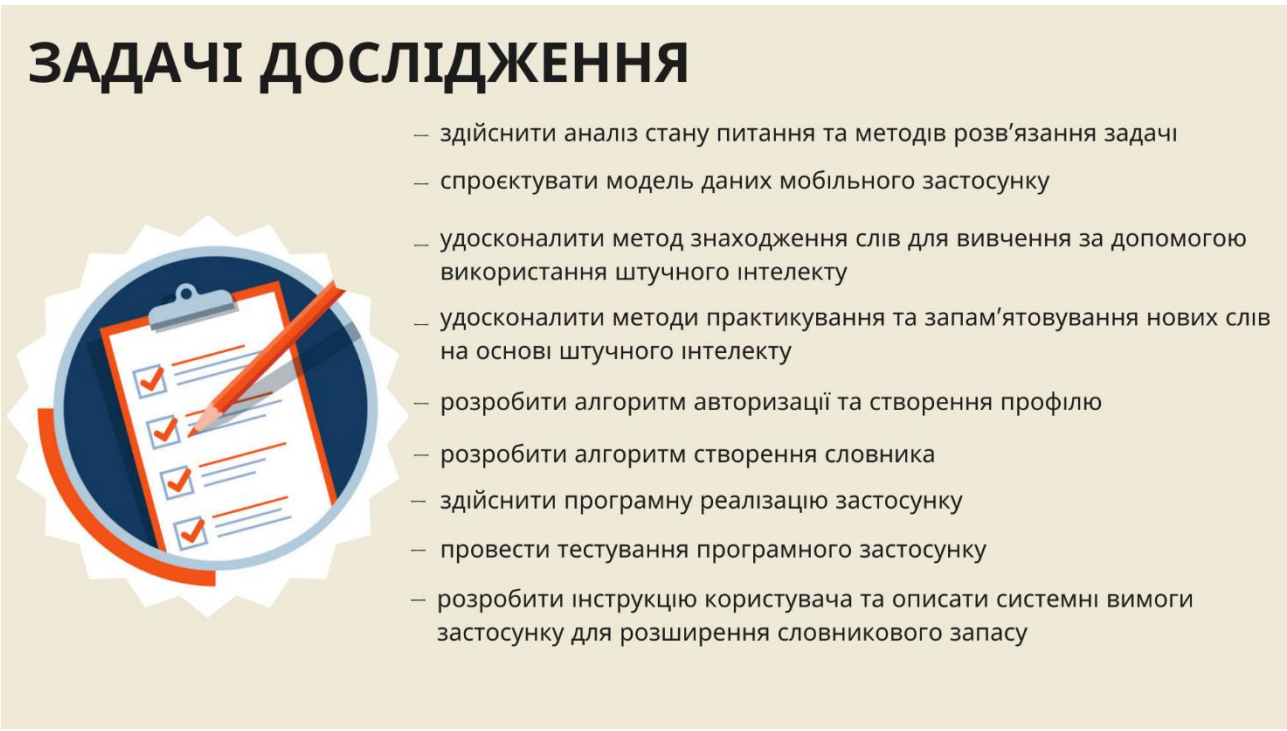



Рисунок Г.4 – Задачі дослідження

# НОВИЗНА ТА ПРАКТИЧНА ЦІННІСТЬ ОТРИМАНИХ РЕЗУЛЬТАТІВ

**Удосконалено** метод знаходження слів для вивчення за допомогою використання штучного інтелекту, який на відміну від відомих, дозволяє підбирати нові слова відносно мови для вивчення та рівня користувача, що дозволило створювати індивідуальні набори під кожного користувача та їх потреби.

**Удосконалено** методи запам'ятовування та практикування нових слів, які, на відміну від відомих, використовують штучний інтелект, що дозволяє генерувати завдання окремо для кожного користувача, які базуються на рівні володіння мовою та словах, які вивчаються.

**Практична цінність.** На основі отриманих теоретичних положень в бакалаврській дипломній роботі запропоновано алгоритми та розроблено програмний мобільний застосунок для розширення словникового запасу.

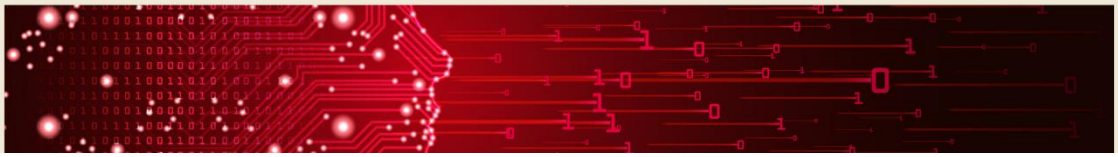


Рисунок Г.5 – Новизна та практична цінність отриманих результатів

# ПОРІВНЯЛЬНИЙ АНАЛІЗ ПРОГРАМ–АНАЛОГІВ

| | Quizlet | Babbel | AnkiApp | JellyVocab |
|---|---|---|---|---|
| Офлайн режим | + | - | + | + |
| Вивчення слів за допомогою написання | + | + | - | + |
| Вивчення слів за допомогою карток | + | - | + | + |
| Вивчення слів у форматі «Ваставка в речення» | - | - | - | + |
| Атоматичне генерування слів за допомогою AI | - | - | - | + |
| Сумарний коефіцієнт | 3 | 1 | 2 | 5 |

Рисунок Г.6 – Порівняльний аналіз програм-аналогів


Рисунок Г.7 – Модель системи


Рисунок Г.8 – Удосконалення методу знаходження слів для вивчення за допомогою використання штучного інтелекту

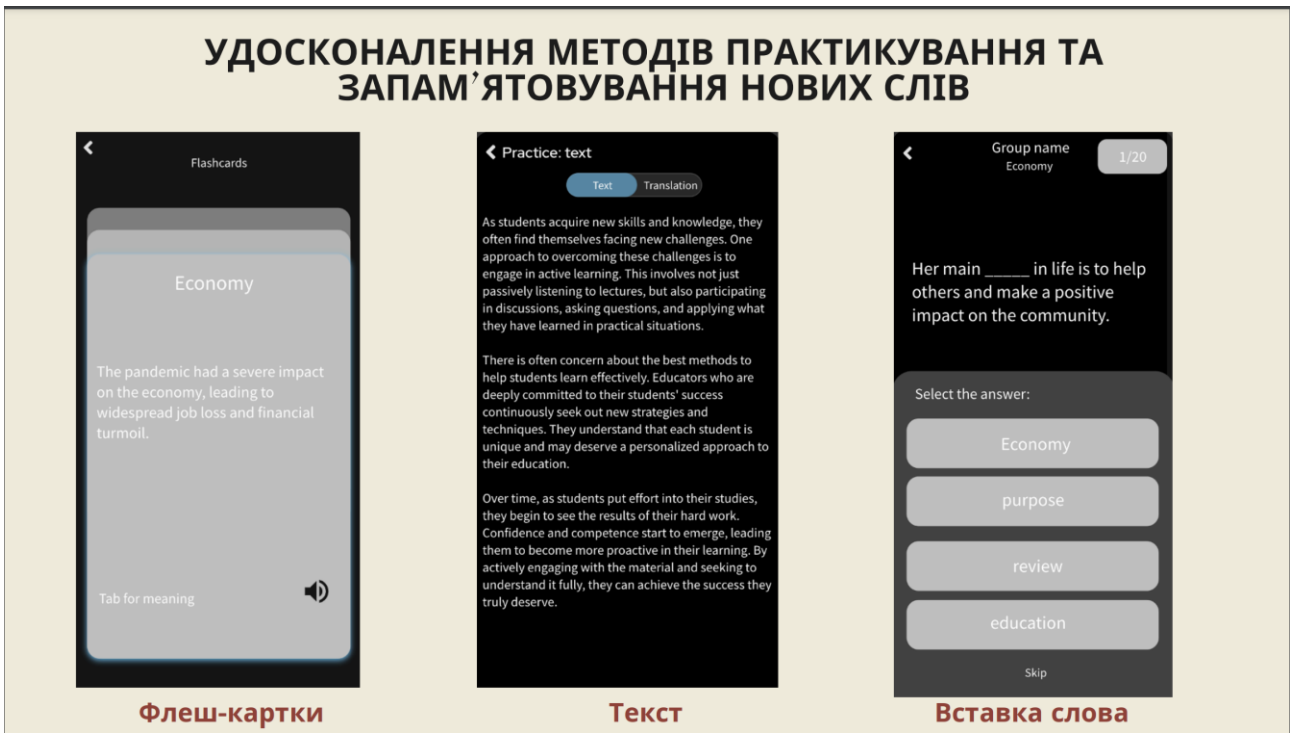



Рисунок Г.9 – Удосконалення методів практикування та запам'ятовування нових слів


Рисунок Г.10 – Блок-схема алгоритму створення словника


Рисунок Г.11 – Блок-схема алгоритму авторизації та створення профілю

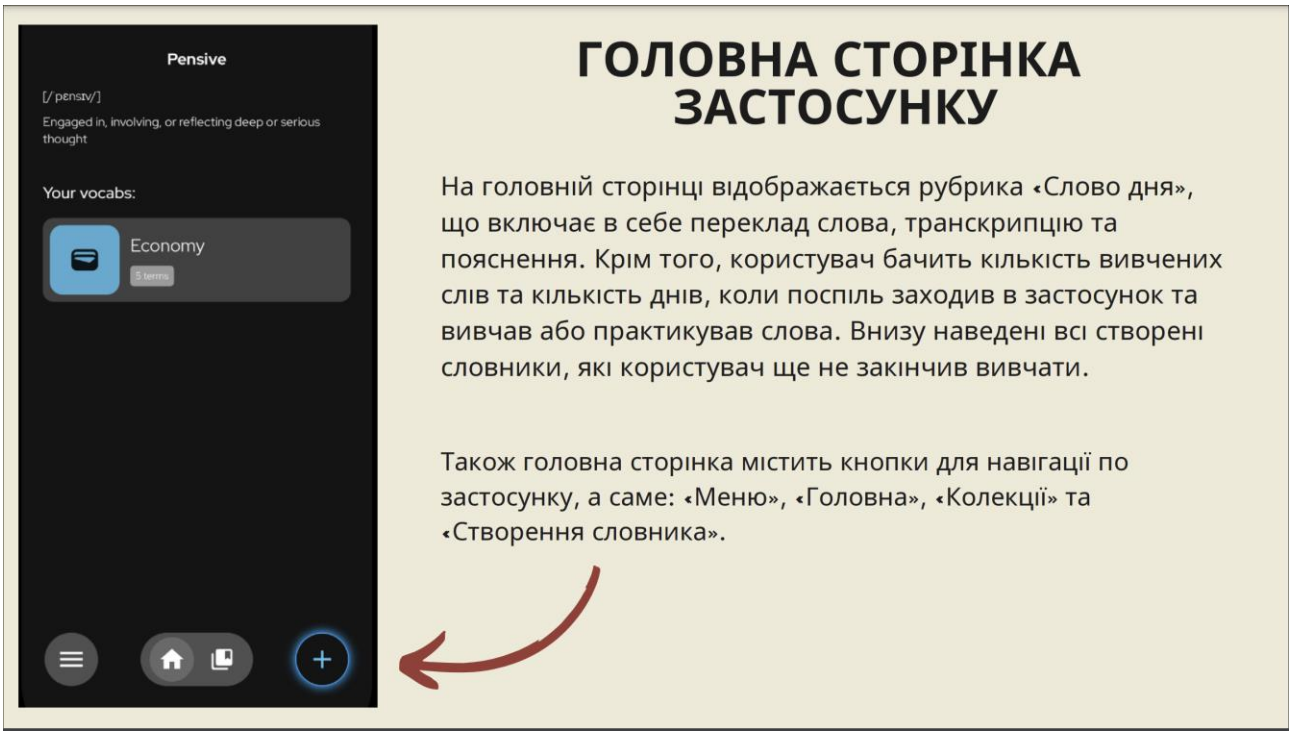



Рисунок Г.12 – Головна сторінка застосунку

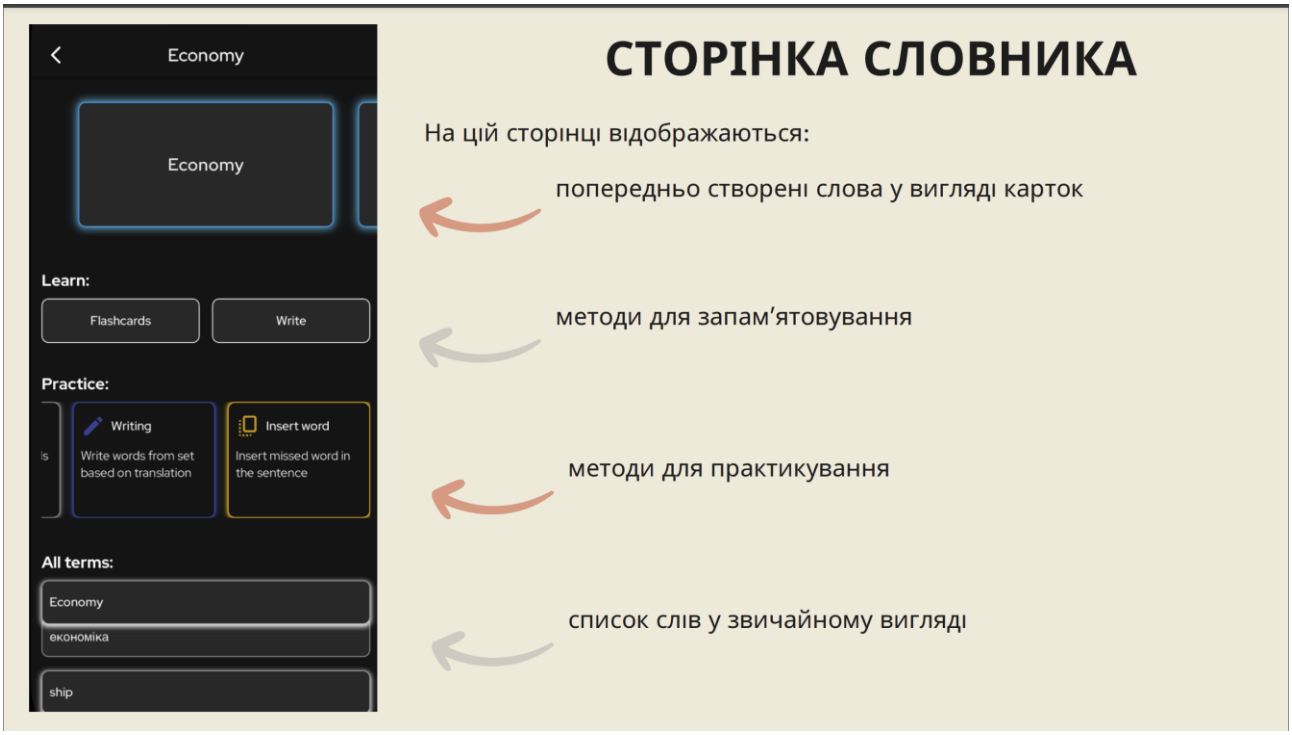



Рисунок Г.13 – Сторінка словника

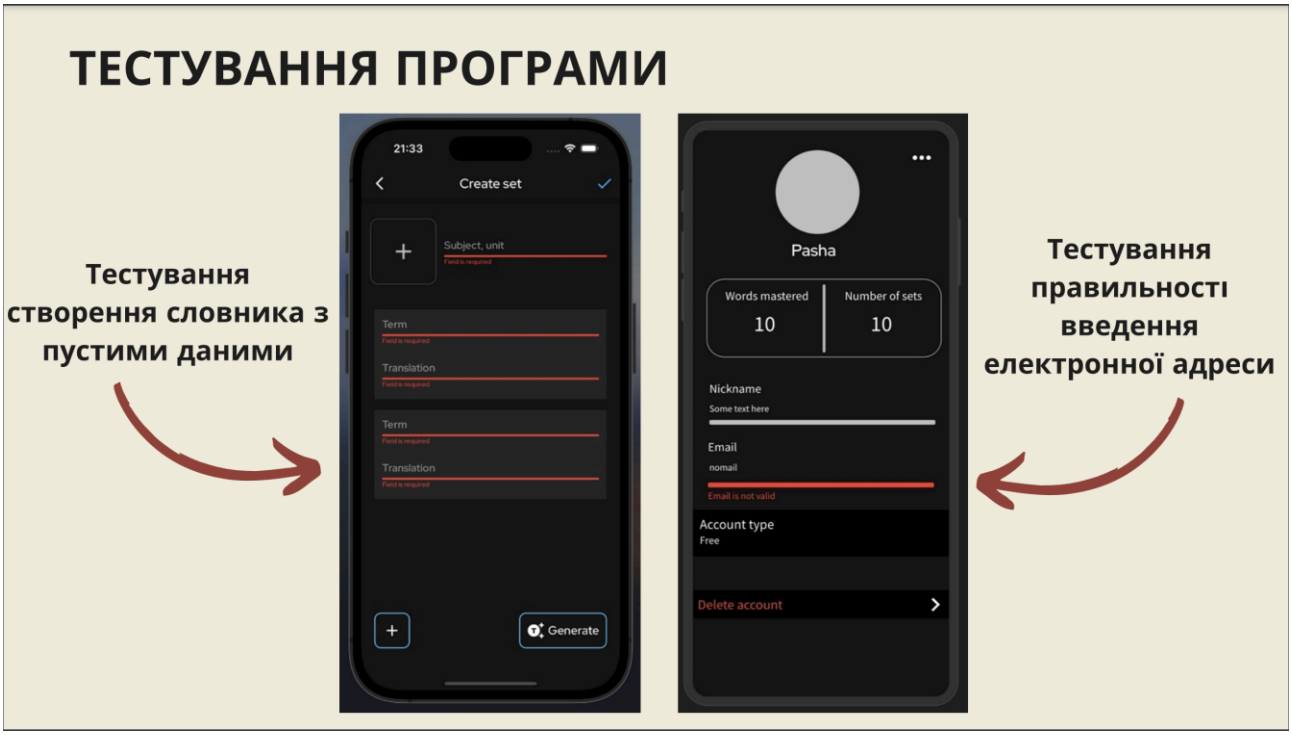



Рисунок Г.14 – Тестування програми


Рисунок Г.15 – Апробація і публікація бакалаврської дипломної роботи

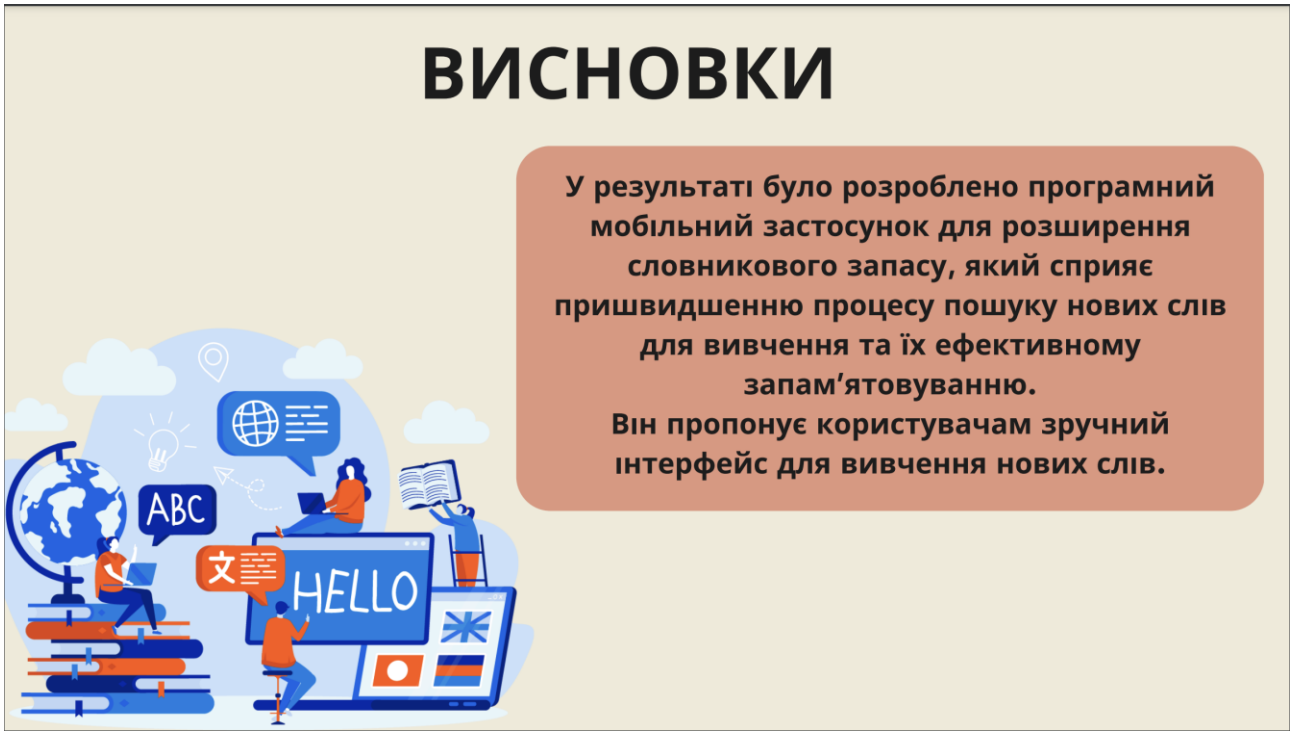



Рисунок Г.16 – Висновки

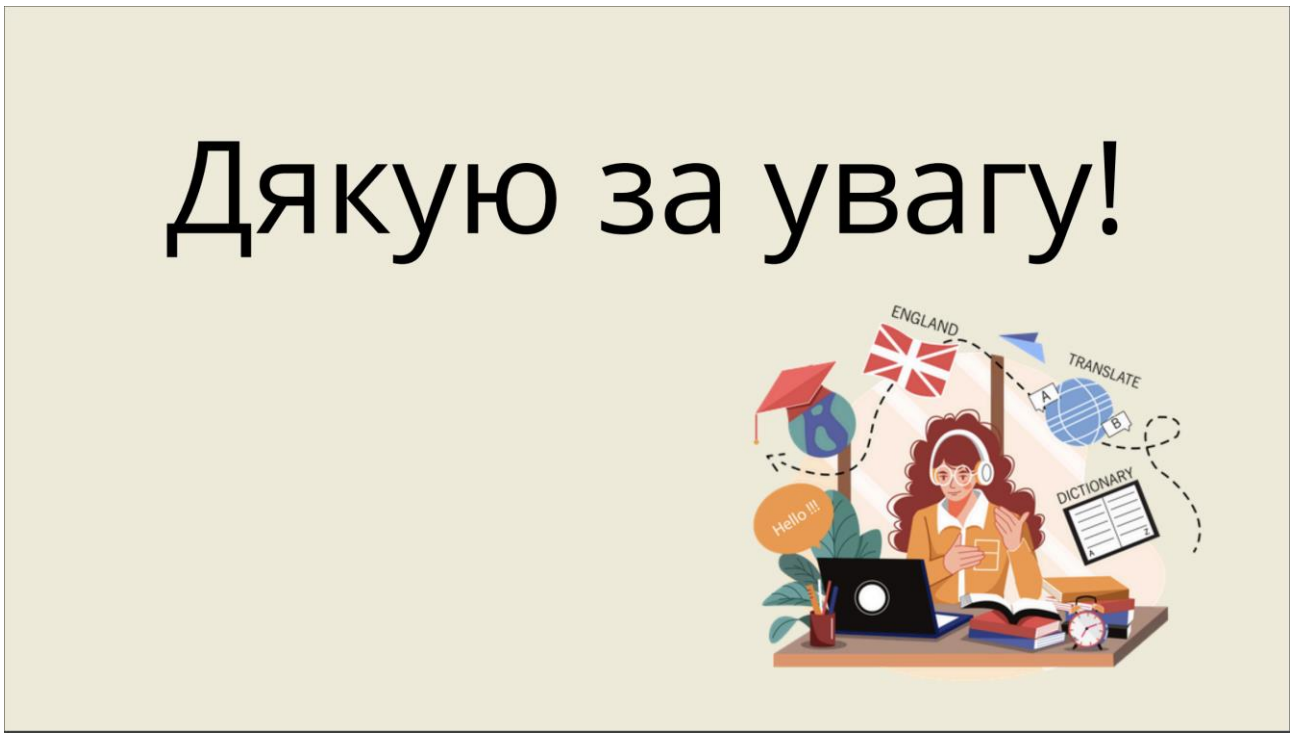



Рисунок Г.17 – Фінальний слайд