ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка мобільного застосунку з тестами з іноземної мови з використанням технологій штучного інтелекту»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Данило ДЗЮБА
(підпис)

Виконав: здобувач вищої освіти групи ПД-44

Данило ДЗЮБА

Керівник: Вячеслав САВІЦЬКИЙ
викладач кафедри ІПЗ

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дзюбі Данилу Романовичу

1. Тема кваліфікаційної роботи: «Розробка мобільного застосунку з тестами з іноземної мови з використанням технологій штучного інтелекту»
керівник кваліфікаційної роботи викладач кафедри ІПЗ Вячеслав САВІЦЬКИЙ, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56.

2. Строк подання кваліфікаційної роботи «02» червня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про методи розробки мобільних застосунків, документація фреймворку Flutter та сервісу Firebase.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Проаналізувати предметну галузь мобільних застосунків для вивчення іноземних мов.
2. Огляд мобільного застосунку з тестами з іноземної мови з використанням технологій штучного інтелекту та аналіз існуючих аналогів, визначення вимог до програмного забезпечення.
3. Огляд та аналіз засобів для реалізації мобільного застосунку.

4. Проектування архітектури мобільного застосунку.
5. Програмна реалізація та тестування мобільного застосунку.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів
2. Вимоги
3. Програмні засоби реалізації
4. Архітектура застосунку
5. Моделювання функціональності
6. Карта розробленого застосунку
7. Екранні форми
8. Відео роботи застосунку
9. Апробація результатів дослідження
10. Висновки

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	25.02.-04.03.2025	
2	Аналіз та дослідження існуючих аналогів	05.03-13.03.2025	
3	Аналіз застосунку з тестами з іноземної мови з використанням ШІ	14.03-17.03.2025	
4	Дослідження та вибір засобів розробки застосунку	18.03-20.03.2025	
5	Проектування програмного забезпечення	21.03-25.03.2025	
6	Програмна реалізація та тестування застосунку	26.03-28.04.2025	
7	Оформлення роботи: вступ, висновки, реферат	29.04-11.05.2025	
8	Розробка демонстраційних матеріалів	12.05-18.05.2025	
9	Попередній захист роботи	19.05-30.05.2025	

Здобувач вищої освіти _____ Данило ДЗЮБА
(підпис)

Керівник
кваліфікаційної роботи _____ Вячеслав САВІЦЬКИЙ
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 59 стор., 4 табл., 32 рис., 20 джерел.

Мета роботи - прискорення вивчення іноземних мов в мобільному застосунку.

Об'єкт дослідження - процес вивчення іноземних мов.

Предмет дослідження - програмне забезпечення для вивчення іноземних мов.

Короткий зміст роботи: В роботі було проаналізовано предметну галузь та методи реалізації для застосунку з тестами з іноземних мов з використанням ШІ. Проведено огляд вже існуючих застосунків для вивчення іноземних мов: Duolingo, Lingvist, Babbel. Розроблено мобільний застосунок та програмно реалізовані основні функціональні можливості: авторизація, вибір мови для тестування, сторінка для проходження тестів, надання пояснень до помилок від ШІ, словник в якому можна дізнатися у ШІ значення того чи іншого слова, можливість голосового вводу тексту в словник та відтворення відповіді від ШІ голосом. В роботі використано фреймворк Flutter та мову програмування Dart для розробки застосунку, в якості моделі ШІ використано llama3.2, в якості бекенду та бази даних використано сервіс Firebase.

Сферою використання застосунку є організація ефективного вивчення іноземних мов користувачами використовуючи ШІ.

КЛЮЧОВІ СЛОВА: FLUTTER, DART, FIREBASE, МОБІЛЬНИЙ ЗАСТОСУНОК, ШТУЧНИЙ ІНТЕЛЕКТ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ - програмне забезпечення

IDE - Integrated Development Environment

SRS - Spaced Repetition System

URL - Uniform Resource Locator

ШІ - штучний інтелект

LLM - Large Language Model

ВНЗ - Вищий Навчальний Заклад

ІТ - Інформаційні Технології

UI - User Interface

UX - User Experience

HTTP - HyperText Transfer Protocol

NLP - Natural Language Processing

API - Application Programming Interface

ВСТУП

Актуальність: У наш час знання іноземних мов відкриває нові освітні та кар’єрні можливості, що зумовлює зростання попиту на зручні засоби навчання. Цифрові технології й мобільні пристрої дали змогу вивчати мови в будь-який час і будь-де.

Особливу роль у цьому відіграють мобільні застосунки, які завдяки штучному інтелекту можуть автоматизувати перевірку завдань і надавати пояснення, роблячи навчання ефективнішим та швидшим.

Завдяки цьому процес вивчення іноземних мов стає ефективнішим та цікавішим для користувача.

Об'єкт дослідження є процес вивчення іноземних мов.

Предмет дослідження є програмне забезпечення для вивчення іноземних мов.

Мета роботи є прискорення вивчення іноземних мов в мобільному застосунку.

Проблема заключається в тому що багато користувачів стикаються з труднощами у вивченні іноземних мов через відсутність персоналізованого підходу.

Більшість мобільних застосунків не надають детального аналізу помилок, що ускладнює процес вивчення а також не запобігає повторенню однотипних помилок.

Методи дослідження: спочатку було проаналізовано існуючі застосунки для вивчення іноземних мов та сформовано початкові функціональні та нефункціональні вимоги.

Стек технологій для розробки додатку склали: фреймворк Flutter для розробки мобільного застосунку, сервіс Firebase як база даних та бекенд проекту, GitHub як систему контролю версій, додаток ollama та модель штучного інтелекту llama3.2 та сервіс Figma для створення прототипу застосунку.

Дослідження реалізації застосунку проводилося безпосередньо під час розробки застосунку.

Наукова новизна визначається використанням технологій штучного інтелекту для перевірки відповідей в тестах від користувачів та надання розгорнутих пояснень помилок.

Практична значущість результатів полягає в можливості використовувати застосунок для ефективного вивчення іноземних мов на смартфона

1 АНАЛІЗ ЗАСТОСУНКУ З ТЕСТАМИ З ІНОЗЕМНОЇ МОВИ З ВИКОРИСТАННЯМ ШІ

1.1 Аналіз предметної галузі застосунків для вивчення іноземних мов

У сучасному світі спостерігається значне зростання інтересу до застосунків, спрямованих на підтримку процесу вивчення іноземних мов. Ця предметна галузь є одним із ключових напрямів у сфері освітніх технологій, що об'єднує мовну підготовку з інноваційними цифровими рішеннями.

Мобільні застосунки для вивчення мов дозволяють користувачам формувати та розвивати навички читання, аудіювання та письма в інтерактивному середовищі. Одним із найбільш ефективних методів перевірки рівня знань є тестування, що широко застосовується у навчальному процесі. Тестові завдання у мовних застосунках дозволяють оцінити рівень володіння лексикою, граматикою, розумінням тексту, а також покращити закріплення вивченого матеріалу.

Серед найпопулярніших представників галузі варто відзначити Duolingo, Lingvist, Babbel та інші, які поєднують традиційні освітні підходи з елементами гейміфікації та мультимедійного контенту. Проте, в останні роки спостерігається тенденція до інтеграції в мовні застосунки технологій штучного інтелекту, що суттєво розширює їх функціональні можливості.

Застосування штучного інтелекту в освітніх мобільних застосунках відкриває нові можливості для автоматичної оцінки відповідей, генерації зворотного зв’язку в реальному часі та проводити аналіз прогресу з метою підвищення ефективності навчального процесу. Використання технологій штучного інтелекту дає змогу реалізувати інтелектуальну підтримку користувача під час проходження тестів, виявляти типові помилки та формувати індивідуальні рекомендації.

Таким чином, предметна галузь застосунків для вивчення іноземних мов, зокрема тих, що реалізують тестування із залученням інструментів штучного інтелекту, є перспективною та суспільно значущою.

1.2 Мобільний застосунок з тестами з іноземної мови

Мобільний застосунок з тестами з іноземної мови - це спеціалізований мобільний застосунок, який надає доступ до різноманітних тестових завдань а також технологій ШІ, які допоможуть в підвищенні мовного рівня користувача. Основна мета мобільного застосунку в наданні ефективного та доступного цілодобово способу перевірки та покращення власних знань з використанням технологій ШІ для пояснення помилок у тестах, а також чат-боту з яким можна попрактикуватися в використанні мови.

Основні компоненти мобільного застосунку включають:

- тести, які структуровані за напрямком та мовою;
- словник, в якому можна дізнатися визначення та приклад використання слов від ШІ;
- чат-бот з яким можна попрактикуватися в використанні мови;
- система оцінювання, де після кожного тесту буде надано оцінку, а на сторінці профілю можна буде побачити середні оцінки по тестам.

Основні цілі застосунку:

- покращення розуміння мови;
- практика використання мови в реальних ситуаціях;
- оцінка прогресу;
- вивчення нових слів.

Основні переваги та недоліки мобільного застосунку перед більш традиційними підходами:

Переваги:

- доступність, а саме можливість навчатися будь-де і будь-коли, при наявності інтернету;

— гнучкість, що надає користувачам можливість навчатися в власному темпі, а також самим обирати пріоритетні теми для вивчення;

— вартість, тобто навчання за допомогою мобільних застосунків може бути повністю безкоштовним, чого не скажеш про традиційні курси.

Недоліки:

— якість контенту, не у всіх мобільних застосунках контент для навчання буде однаково корисним;

— відсутність контакту з людьми може бути доволі серйозною проблему, адже тільки з іншою людиною можливо справді потренуватися в мовленні на іноземній мові;

— технічні проблеми, які можуть виникати час від часу, і можуть завадити навчанню;

— самоорганізація, адже через відсутність чіткого графіку навчання, може бути доволі складно ефективно організувати його самому.

1.3 Цільова аудиторія

Застосунок орієнтований на широке коло користувачів потреби яких можуть різнитися. До основних категорій належать:

— ентузіасти - люди які бажають вивчати іноземні мови для своїх професійних чи особистих цілей. Їх мета - розширити свої знання та навички в іноземних мовах;

— студенти мовних спеціальностей - це студенти ВНЗ та інших навчальних закладів. Їх мета - поглиблене вивчення обраних мов для іспитів чи професійної діяльності;

— особи що готуються до мовних іспитів - студенти та ентузіасти, які прагнуть скласти мовні іспити. Їх мета - підвищення мовного рівня для успішного складання іспитів.

1.4 Дослідження існуючих аналогів

На сьогоднішній день найпопулярнішими застосунками для вивчення мов є Duolingo, Lingvist, Babbel. Розглянемо кожну з них.

1.4.1 Duolingo

Duolingo - це популярний мобільний застосунок для вивчення іноземних мов. В ньому доступно більше 40 мов для вивчення. Застосунок надає змогу безкоштовно вивчати іноземні мови. Підхід до навчання складається з коротких інтерактивних уроків та системи досягнень для підтримання зацікавленості.

Основні функції Duolingo:

- система цілей та рівнів, а саме можливість встановлювати цілі по кількості пройдених рівнів на день.
- розділ на тем за тематикою;
- інтервальне повторення (SRS) - це система час від часу пропонує перевірити знання по вже пройденим темам;
- гейміфікація процесу навчання через мотиваційну систему, яка складається з щоденних серій входу, бали та нагороди за досягнення.
- персоналізація навчання - це система, яка автоматично адаптується під рівень користувача;
- система ліг, яка дозволяє користувачам відчути елемент змагання з іншими користувачами.

Недоліки Duolingo:

- Duolingo більшою мірою орієнтований на початківців, тож для користувачів середнього та вище рівня буде мати обмежену користь;
- слабкий розвиток мовлення, адже завдання на вимову в більшості доволі примітивні. В результаті можна пройти курс але не вміти говорити в реальних ситуаціях;
- багато вправ зводяться до механічного заучування правильних відповідей.

На рисунку 1.1 показано екран тестування в мобільному застосунку Duolingo, а на рисунку 1.2 показано сторінку вибору уроків та вправ в Duolingo.

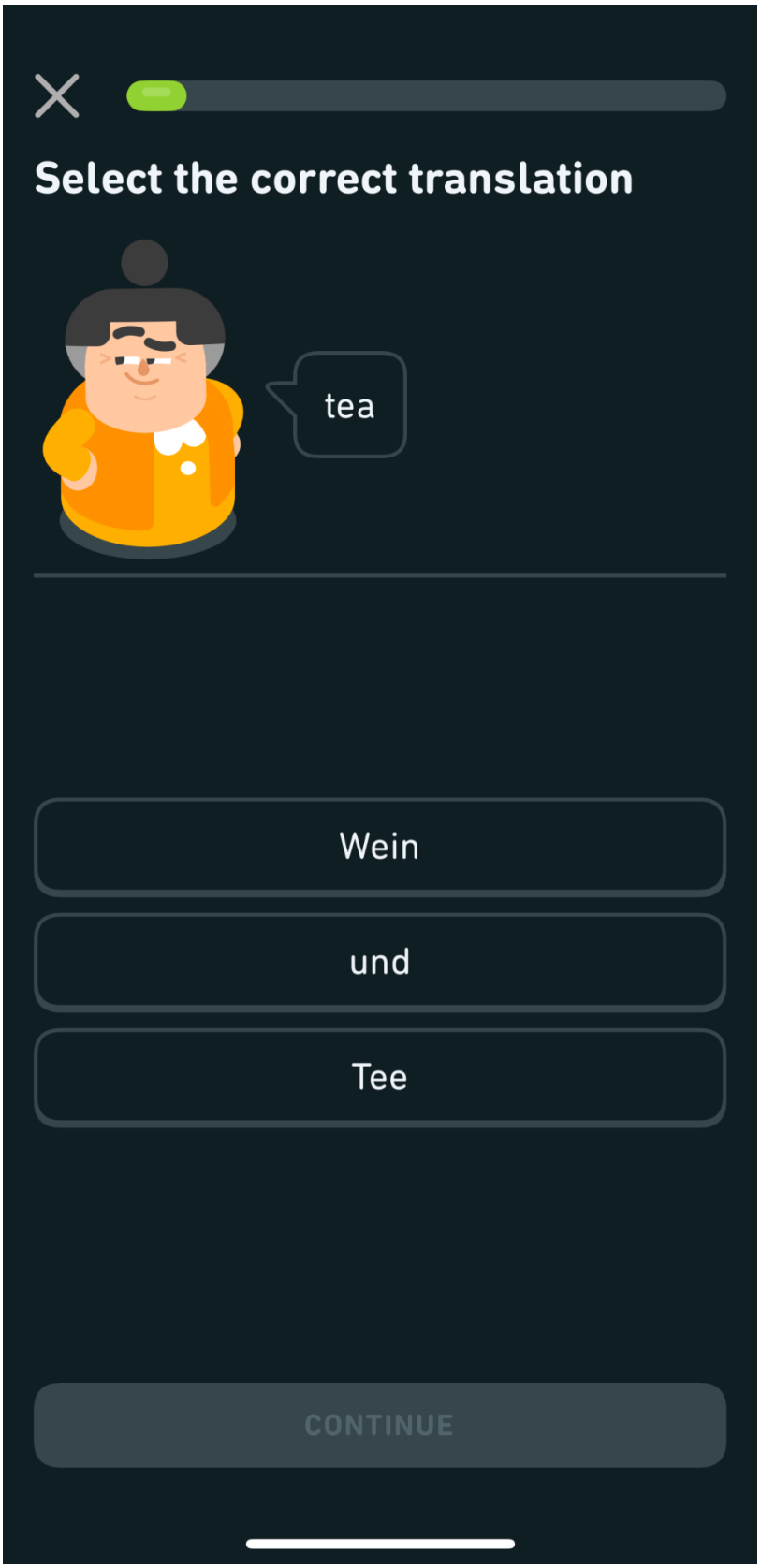


Рис. 1.1 Сторінка тестування Duolingo

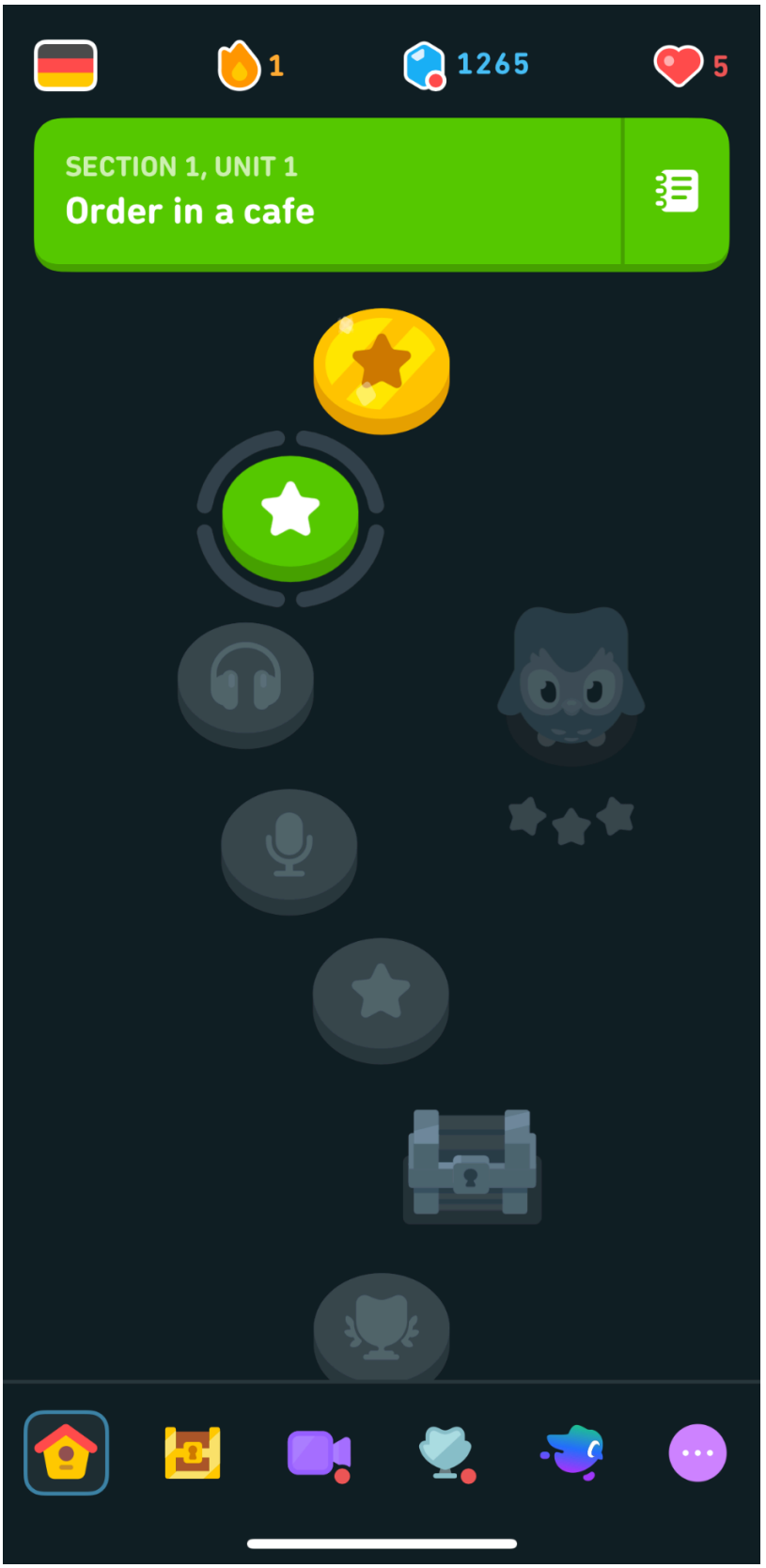


Рис. 1.2 Сторінка уроків та вправ Duolingo

1.4.2 Lingvist

Lingvist - це мобільний та веб-додаток для вивчення іноземних мов з акцентом на розвиток словникового запасу. Спочатку йде вивчення найпоширеніших слів, які охоплюють 80% повсякденних випадків. Lingvist використовує алгоритм інтервального повторювання, який гарантує, що ви вивчатимете потрібні слова саме тоді коли потрібно [1]. Наразі доступні мови для вивчення: англійська, іспанська, французька, німецька, японська, корейська, італійська, естонська, нідерландська, польська та шведська.

Основні функції Lingvist:

- гнучкі налаштування, які дозволяють під себе налаштувати курс або набір слів для вивчення;
- спеціалізовані курси, такі як бізнес англійська або підготовка до іспитів (TOEFL, IELTS);
- контекстне навчання, тобто слова подаються в контексті, що сприяє їх вивченню;
- фокус на статистиці, а саме детальна аналітика прогресу (точність відповідей, кількість вивчених слів).

Недоліки Lingvist:

- слабкий розвиток граматики, а саме вправи в основному зосереджені на словниковому запасі;
- мінімальна геймифікація, немає системи ліг чи візуальних нагород;
- Відсутня розмовна практика.

На рисунку 1.3 показано сторінку з питаннями в мобільному застосунку Lingvist, а на рисунку 1.4 сторінку вибору курсів.

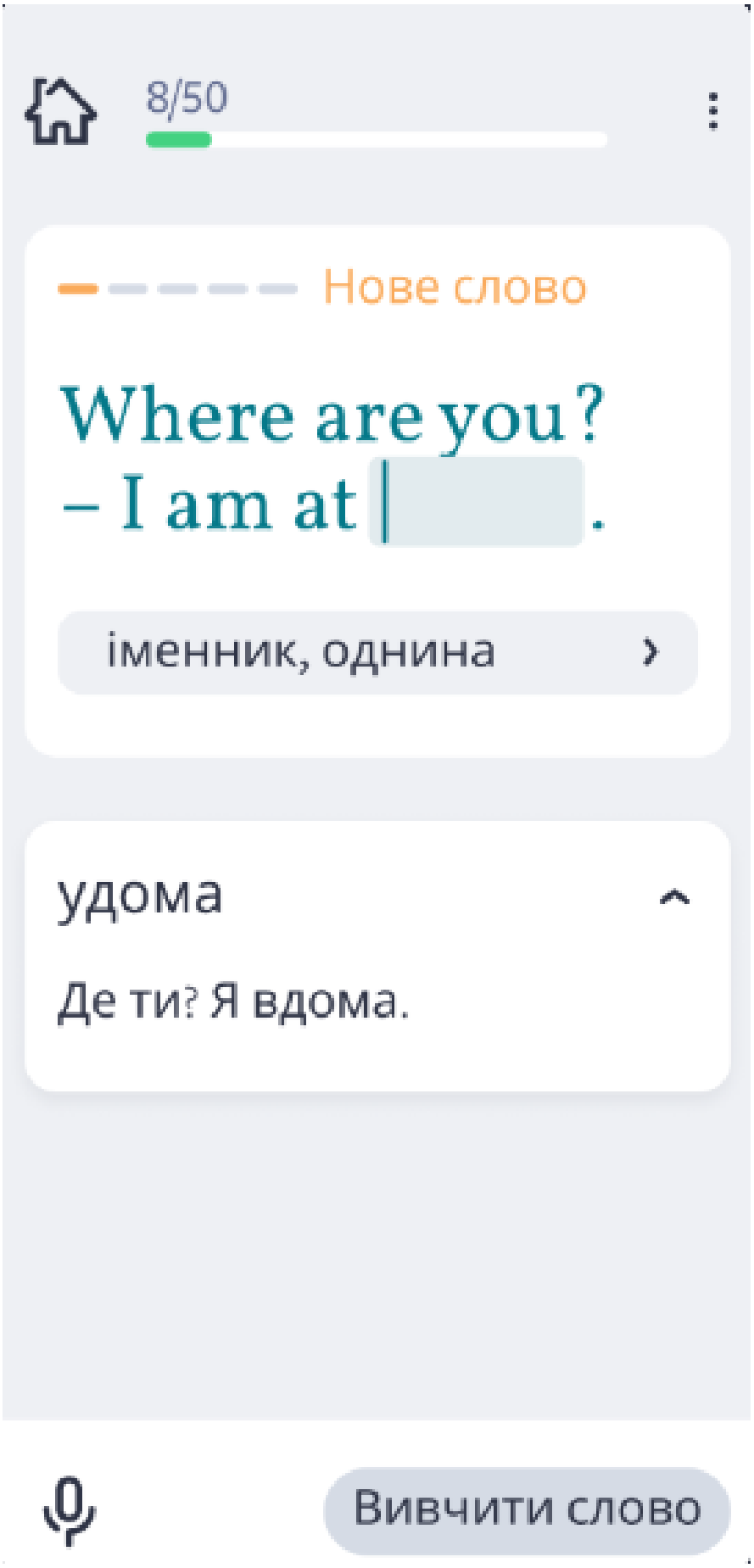


Рис. 1.3 Сторінка з питанням в Lingvist

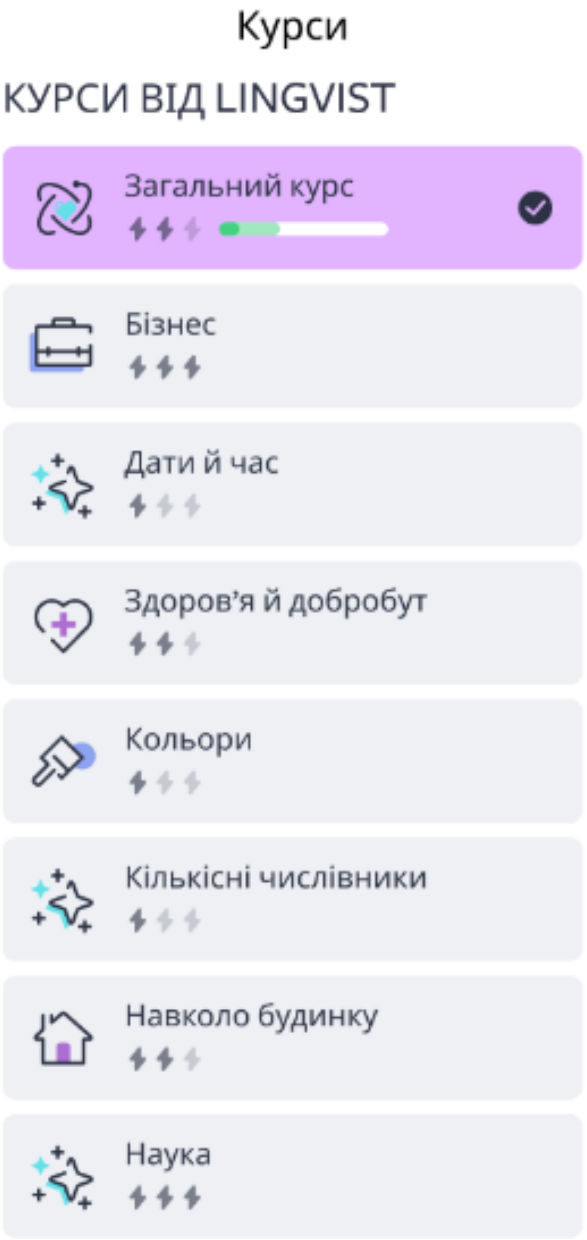


Рис. 1.4 Сторінка вибору курсів в Lingvist

1.4.3 Babbel

Babbel - це платна платформа, яка пропонує курси для вивчення іноземних мов через мобільний застосунок та веб-сайт. Одна з найбільших переваг Babbel в фокусі на моделюванні реальних ситуацій для вивчення мови. Курси Babbel розроблені у такий спосіб, щоб допомогти користувачам розвивати навички англійської мови, необхідні для роботи в IT-галузі, такі як технічна термінологія, комунікація з колегами, участь у відкритих дискусіях та інші аспекти, що є важливими для професійної комунікації [4]. Наразі доступні

для вивчення 13 мов більшість з яких європейські, але також доступні турецька та індонезійська.

Основні функції Babbel:

— структуровані уроки,а саме короткі уроки (10-15 хвилин), що зосереджені на реальних ситуаціях;

— розмовна практика, тобто вправи з вимовою за допомогою технології розпізнавання мовлення;

— інтервальне повторення (SRS) - це система час від часу надає матеріал для закріплення знань;

— Babbel Live, яка надає онлайн-уроки з викладачами у малих групах.

Недоліки Babbel:

— обмежена кількість мов серед доступних 13 мов переважно європейські;

— основна увага на початківцях, а саме мало або взагалі немає навчального матеріалу поза рівнями A1-B1;

— однотипність вправ, тобто вправи доволі однотипні в довгостроковій перспективі і втрачають свою ефективність.

На рисунку 1.5 показано екранні форми мобільного застосунку Babbel.

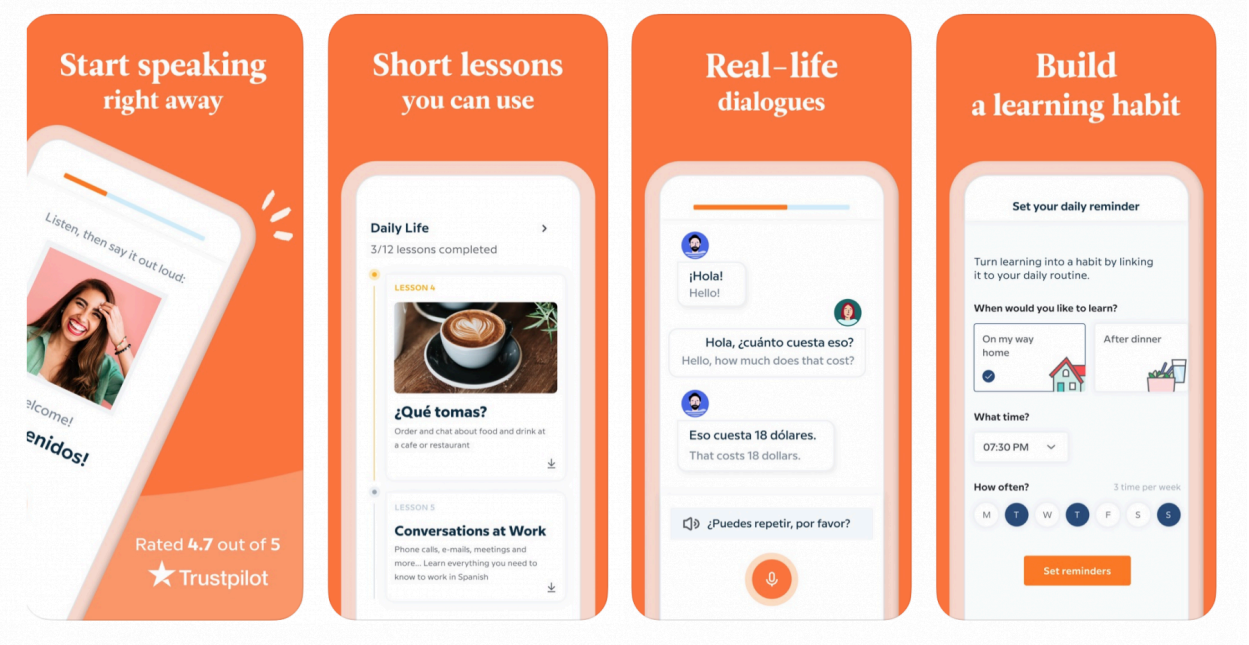


Рис. 1.5 Екранні форми застосунку Babbel

В таблиці 1.1 наведено результати аналізу застосунків для вивчення іноземних мов та їх порівняння.

Таблиця 1.1

Таблиця існуючих застосунків для вивчення іноземних мов та їх порівняння

Показник	**Duolingo**	**Lingvist**	**Babbel**
Доступність на платформах	Web, IOS, Android	Web, IOS, Android	Web, IOS, Android
Обмеження без підписки	Доступ лише до базових функцій	Доступ лише за підпискою	Для українців безкоштовно Німецька, польська, англійська
Цільова аудиторія	Початківці та люди, що готуються до іспитів	Початківці, студенти та ентузіасти	Початківці, студенти та мандрівники
Прогрес та статистика	Система досягнень	Детальна статистика успішності	Рівень володіння мовами, активність
Мотиваційна система	Бали, досягнення, щоденні серії	Щоденні цілі, Графіки прогресу та рівні	Щоденні серії, нагороди та інтерактивні завдання

Продовження таблиці 1.1

Таблиця існуючих застосунків для вивчення іноземних мов та їх порівняння

Показник	**Duolingo**	**Lingvist**	**Babbel**
Особливості	Інтерактивний підхід, гейміфікація	Індивідуальний підхід, упор в практичність	Науковий підхід до курсів, розбиття курсів за рівнем слухача
Недоліки	Відсутність ряду функцій в безкоштовній версії	Відсутня безкоштовна версія. Лише пробний період	Безкоштовно доступні лише Німецька, польська, англійська

В таблиці 1.2 наведено порівняння розробленого проекту з аналогами.

Таблиця 1.2

Таблиця порівняння існуючих застосунків для вивчення іноземних мов та розробленого проекту

Показник	**Duolingo**	**Lingvist**	**Babbel**	Розроблений проект
Платформа	Веб-сайт, IOS, Android	Веб-сайт, IOS, Android	Веб-сайт, IOS, Android	Android
Чат-бот	+	+	+	+
ШІ словник	-	-	-	+

Продовження таблиці 1.2

Таблиця порівняння існуючих застосунків для вивчення іноземних мов та розробленого проекту

Показник	**Duolingo**	**Lingvist**	**Babbel**	Розроблений проект
Інтелектуальне пояснення помилок в тестах	-	-	-	+

1.5 Визначення вимог до застосунку

Проаналізувавши тематику застосунку, його цільову аудиторію та аналоги. Було визначено наступні вимоги:

Функціональні вимоги:

- реєстрація та автентифікація користувачів;
- вибір мови для тестування;
- проходження тестів у форматі вибору правильної відповіді;
- розбивка тестів на категорії за темами;
- використання технологій штучного інтелекту для аналізу відповідей та надання пояснень до помилок;
- реалізувати чат-бот для практики переписки;
- створити словник, який за допомогою штучного інтелекту буде надавати пояснення слова.

Нефункціональні вимоги:

- сумісність із операційною системою Android;
- час завантаження сторінок до 3 секунд;
- забезпечення захисту персональних даних користувачів за допомогою використання Firebase Authentication;
- інтерфейс мобільного застосунку повинен бути адаптивним до розмірів екрану.

2 ЗАСОБИ РОЗРОБКИ ЗАСТОСУНКУ

Для розробки мобільного застосунку з тестами з іноземної мови з використанням технологій штучного інтелекту потрібно спроектувати користувацький інтерфейс, обрати базу даних та написати сам застосунок. Платформою для розробки було обрано Android, середовищем для розробки Android Studio, яка підтримує фреймворк для розробки Flutter. В якості бекенду було використано сервіс Firebase. В якості штучного інтелекту було обрано модель від компанії Meta, а саме llama3.2, а також платформа для локального запуску цієї моделі Ollama.

2.1 Сервіс для розробки інтерфейсів Figma

Figma - це популярний онлайн сервіс для розробки UI/UX застосунків для будь-якої платформи. Ключовою особливістю цього сервісу є повна веб-інтеграція, яка дозволяє користувачам працювати з сервісом без необхідності завантажувати застосунок. Ще однією особливістю Figma є можливість роботи над одним проектом декількох людей, незалежно від їх фізичного розташування.

Figma включає в себе зручні та ефективні інструменти для створення інтерактивних прототипів. Це дозволяє UI/UX дизайнерам швидко створювати прототипи з анімаціями та взаємодією, і тестувати їх прямо в середовищі Figma.

Такий підхід дозволяє мінімізувати кількість змін та правок до UI/UX складової проекту на етапі його розробки.

2.2 Середовище розробки Android Studio

Інтегроване середовище розробки (IDE) - це ПЗ, що надає програмістам в одному місці всі необхідні для розробки інструменти. Інструментами, що

зазвичай включають в себе ICP: текстовий редактор, компілятор/інтерпретатор, дебагер. Можна сказати, що воно є свого роду «робочим столом» для розробників, де вони можуть зосередитися на написанні коду без необхідності перемикатися між різними програмами та інструментами [6].

Android Studio - це інтегроване середовище розробки для розробки мобільних застосунків, перша стабільна версія була представлена в грудні 2014 року компанією Google. Android Studio прийшла на заміну плагіну Android Development Tools для Eclipse. Android Studio розробляється на основі IntelliJ IDEA.

Основні особливості Android Studio:

- емулятор мобільних девайсів;
- уніфіковане середовище для розробки Android застосунків;
- широка підтримка засобіва для тестування застосунків;
- lint інструменти для виявлення проблем продуктивності, сумісності версій і т.д.

На рисунку 2.1 показано інтерфейс інтегрованого середовища розробки Android Studio.

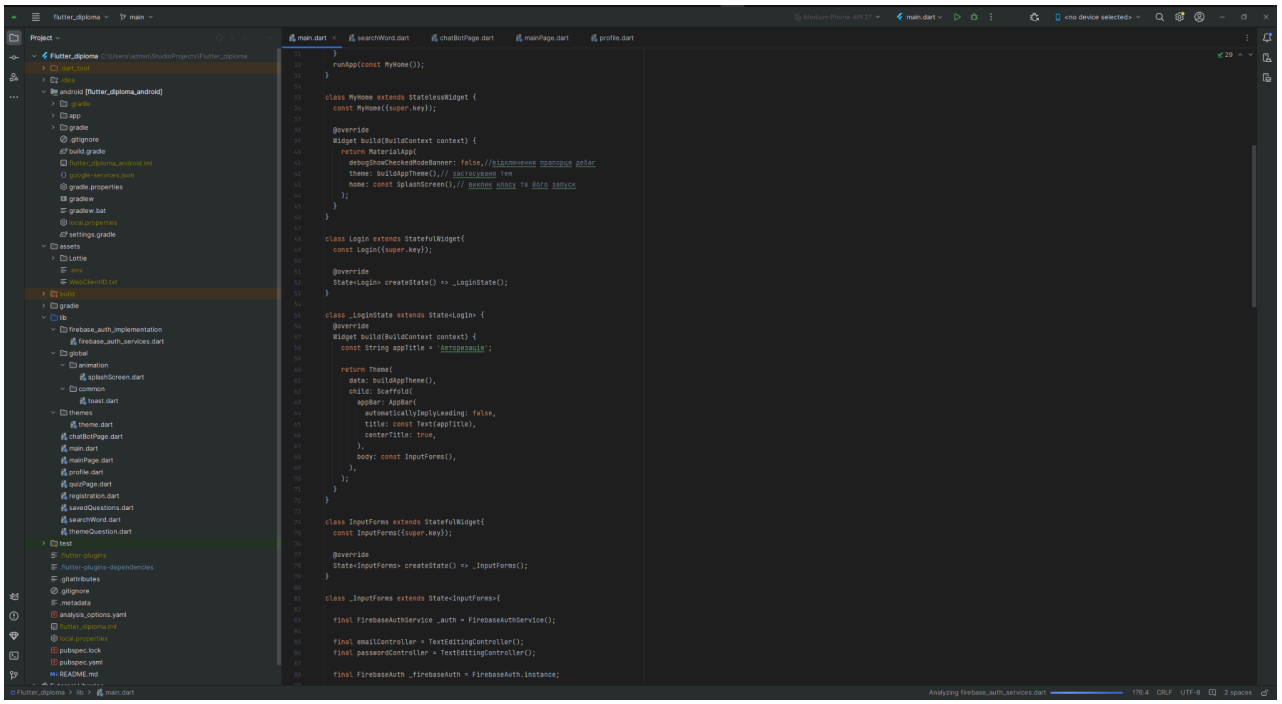


Рис. 2.1 Інтерфейс Android Studio

2.3 Мова програмування Dart

Мова програмування - це штучна мова за допомогою якої записують алгоритми у формі придатній для їх виконання комп’ютером. Вона надає розробникам інструменти для написання програм, управління даними та апаратними ресурсами комп’ютера.

Dart - це мова програмування з відкритим вихідним кодом. Вона є повністю об’єктно-орієнтованою з C подібним синтаксисом. Створена компанією Google в 2011 році і пізніше визнана стандартом ECMA.

Основні особливості Dart:

- відкритий вихідний код;
- платформно незалежна: в Dart є своя віртуальна машина, яка дозволяє запускати його код на будь-якому пристрої;
- об’єктно-орієнтована;
- асинхронна;
- строго типізована.

2.4 Фреймворк для мобільної розробки Flutter

Фреймворк - це набір інструментів, правил та бібліотек. Зазвичай він є структурою, що визначає, як компоненти застосунку повинні взаємодіяти між собою, які шаблони використовувати під час розробки та як взаємодіяти із зовнішніми ресурсами.

Flutter - це фреймворк з відкритим кодом для створення застосунків для платформ Android, iOS та Web. Розроблений компанією Google.

Flutter складається з:

- рушія, який використовує власний рушій написаний на C++ з використанням графічної бібліотеки Google Skia. Він також використовує SDK платформ Android або iOS;

— базової бібліотеки, яка написана на Dart, та слугує для побудови програм та взаємодії з рушієм;

— віджетів, які відповідають за інтерфейс у Flutter. Віджет у Flutter являє собою незмінний об’єкт, що описує частину інтерфейсу користувача.

На поточний час Flutte має два набори віджетів:

— Material Design від Google;

— Cupertino, що імітує дизайн Apple.

2.5 Бекенд сервіс Firebase

Firebase - це сервіс від компанії Google, який надає розробникам набір хмарних сервісів для створення, підтримки та розвитку мобільних та веб застосунків.

Firebase включає в себе:

— Analytics, що надає аналітику по більш ніж 500 різних подіях. Ця статистика дозволяє приймати найкращі рішення щодо розвитку застосунку, виходячи з поведінки користувача;

— Authentication, яка спрощує для розробників захист систем автентифікації. Також цей інструмент підтримує багато постачальників автентифікації, таких як: Google, GitHub, Facebook та інші;

— Cloud Messaging інструмент, який дозволяє компаніям надсилати і отримувати повідомлення на iOS, Android та web безкоштовно;

— Realtime Database це хмарна NoSql база даних, яка дозволяє зберігати та синхронізувати дані між користувачами в режимі реального часу;

— Crashlytics інструмент для відстеження збоїв у реальному часі;

— Performance надає розробникам розуміння на які аспекти застосунку варто звернути увагу і в першу чергу покращити;

— Test Lab це інструмент для тестування застосунків.

2.6 Система контролю версій GitHub

Система контролю версій - це програмний інструмент, що дозволяє відстежувати зміни в коді або документах, зберігаючи історію змін.

Існує 3 види систем контролю версій:

- локальний, в якому всі зміни зберігаються в новому окремому файлі на комп'ютері;
- централізований, який схожий на локальний, але всі файли зберігаються на сервері, а доступ до них надається через спеціальні протоколи;
- децентралізований, на відміну від централізованого, кожен з користувачів отримує весь репозиторій, а не тільки запитувані файли. Тож навіть якщо з сервера буде видалено репозиторій, його можна буде відновити за допомогою локального.

GitHub - це один з найбільших вебсервісів для спільної розробки програмного забезпечення. Він використовує Git для забезпечення управління версіями. За допомогою GitHub розробники можуть одночасно в одному проекті писати код, відстежувати зміни та вирішувати проблеми, що виникають в процесі розробки.

2.7 Ollama платформа для роботи з LLM

LLM - це моделі штучного інтелекту, в галузі обробки природної мови (NLP), які навчені на великих масивах даних з різноманітних джерел (інтернет, книги, наукові статті тощо). Завдяки цьому вони здатні вирішувати широкий спектр завдань, а саме: машинний переклад, аналіз тексту на помилки, генерацію тексту, пошук інформації, програмування і багато чого ще без потреби в налаштуванні під кожну окрему задачу. Але попри всі ці плюси вони не є загальним штучним інтелектом, адже в них є проблеми в розумінні контексту, логічному мисленні та точності фактів.

llama3.2 - це легка текстова модель була розроблена компанією Meta, вона оптимізована для виконання завдань, таких як узагальнення тексту, перевірка його на помилки та виконання інших маніпуляцій з текстом.

Основними перевагами цієї моделі є:

— компактність моделі, а саме розмір моделі 2 гігабайта;

— висока довжина контексту в цілих 128 тисяч токенів, завдяки чому вона здатна обробляти довгий текст;

— відкритий вихідний код, що означає, що цю модель може використовувати будь-який розробник безкоштовно, і що найголовніше її можна адаптувати під потреби проекту;

— вартість, а саме той факт що ця модель є повністю безкоштовною для завантаження.

Для роботи з моделями ШІ в проектах є два основних підходи. Використовувати модель через API або ж запускати її локально. В таблиці нижче надано порівняння цих двох підходів. В таблиці 2.1 наведено порівняння арі моделі та локальної моделі.

Таблиця 2.1

Порівняльна таблиця локальних моделей ШІ та з доступом через API

Критерій	**API модель**	**Локальна модель**
Спосіб використання	Через запити до API стороннього сервісу (наприклад OpenAI)	Модель запускається безпосередньо на комп’ютері чи власному сервері
Налаштування та запуск	Проста інтеграція через REST API або SDK	Необхідно встановити модель, інсталювати платформу для запуску (наприклад Ollama)

Продовження таблиці 2.1

Порівняльна таблиця локальних моделей ШІ та з доступом через API

Продуктивність	Зазвичай висока, але залежить від стабільності з’єднання з сервером провайдера моделі	Залежить лише від продуктивності пристрою на якому запускається
Вимоги до обладнання на якому запускається	Достатньо мати з’єднання з сервером провайдера моделі	Бажано мати потужний процесор, відеокарту та достатній обсяг оперативної пам’яті
Конфіденційність	Дані передаються через мережу на сторонні сервіси (можливі ризики витоку даних)	Повна конфіденційність
Гнучкість	Обмежена політикою провайдера моделі	Повний контроль: можна кастомізувати та донавчити модель під певні задачі
Залежність від зовнішніх сервісів	Робота припиниться без доступу до API чи вичерпанні запитів	Відсутня
Оновлення моделей	Вся відповідальність лежить на провайдері моделі	Потрібно самостійно оновлювати модель

Продовження таблиці 2.1

Порівняльна таблиця локальних моделей ШІ та з доступом через API

Критерій	**API модель**	**Локальна модель**
Ціна	Оплата йде за запити	Безкоштовно

Ollama - це платформа для локального запуску моделей ШІ (LLM), розроблена компанією Meta. Ollama спрощує інсталяцію та використання моделей ШІ. адже все зводиться до завантаження моделі з сайту Ollama та платформи і декількох команд в командному рядку. Основні команди для використання платформи Ollama:

- ollama pull <model> команда яка завантажує обрану модель на користувацьку систему;
- ollama show <model> команда яка показує основну інформацію про обрану модель, а саме її конфігурацію і дату випуску;
- ollama list: виводить список всіх завантажених моделей;
- ollama ps: виводить список всіх запущених моделей;
- ollama run <model> команда яка запускає обрану модель;
- ollama stop <model> команда яка зупиняє роботу обраної моделі;
- ollama rm <model> команда яка видаляє обрану модель з системи;
- ollama help команда яка надає допомогу з будь-якою командою.

3 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура застосунку

На основі обраних технологій було розроблено архітектуру системи за допомогою архітектурного патерну MVC (Model-View-Controller) представлена на рисунку 3.1.

MVC (Model-View-Controller) - це архітектурний патерн, який розділяє мобільний застосунок на три компоненти: модель (model), представлення (view) та контроллер (controller).

- модель відповідає за реалізацію бізнес-логіки в застосунку та управління даними;
- представлення відповідає за користувацький інтерфейс;
- контролер відповідає за взаємодію між моделлю та видом.

Основні переваги використання патерну MVC:

- розділення відповідальності, що спрощує підтримку та тестування коду;
- легша підтримка та масштабованість, через те що зміни в логіці не впливають на вид і навпаки;
- можливість повторного використання коду.

Запропонована архітектура охоплює як клієнтську, так і серверну частину розробленого застосунку та детально описує принципи його функціонування. Завдяки впровадженню сучасних технологій, архітектура забезпечує ефективну обробку запитів та запитів і даних, а також зручну взаємодію з користувачем. Клієнтська частина реалізована з використанням фреймворка Flutter та мови програмування Dart. За функціонування серверної частини відповідає сервіс Firebase, який забезпечує реалізацію бізнес-логіки та зберігання даних.

Окрім цього, у застосунку інтегровано модель штучного інтелекту llama3.2, яка виконує функції надання пояснень до помилок користувача, чат-бота та електронного словника.

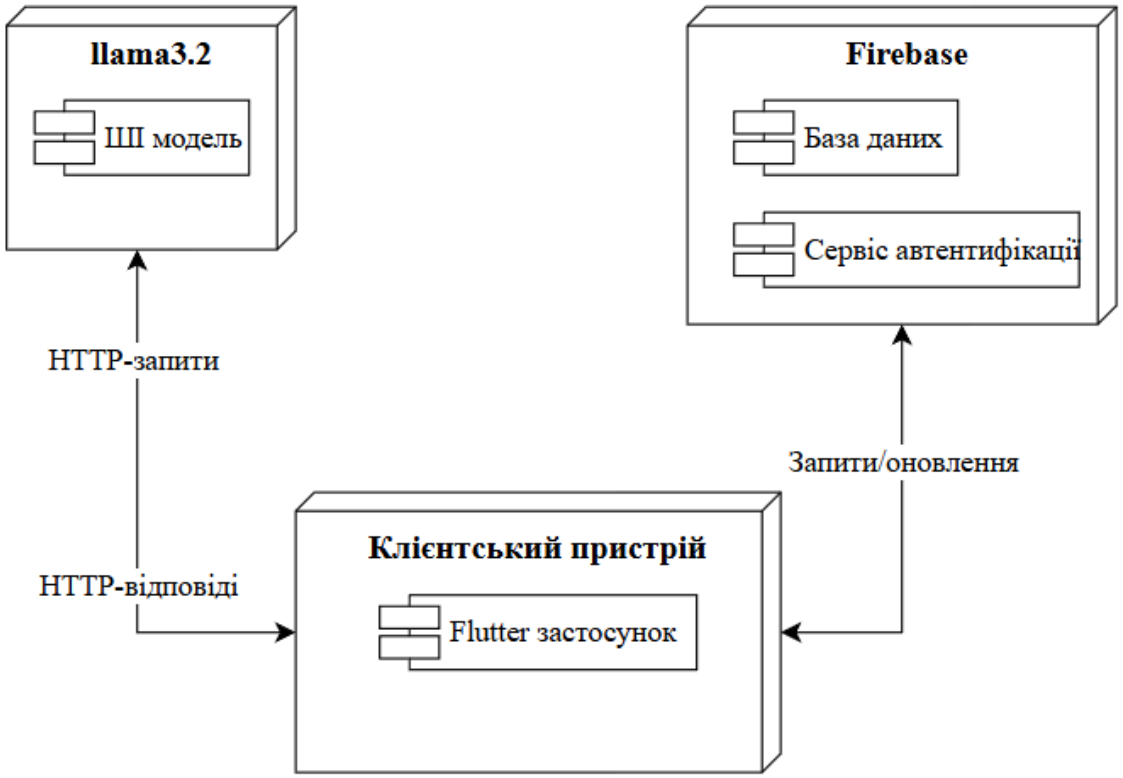


Рис. 3.1 Діаграма розгортання мобільного застосунку

3.2 Клієнтська частина

В основі застосунку лежить фреймворк Flutter, який використовується для створення мобільного застосунку під операційну систему Android. Flutter забезпечує декларативний та компонентний підхід до побудови інтерфейсу користувача, де кожен віджет відповідає за певний інтерфейс чи функціональність. Нижче на рисунку 3.2 наведено карту мобільного застосунку.

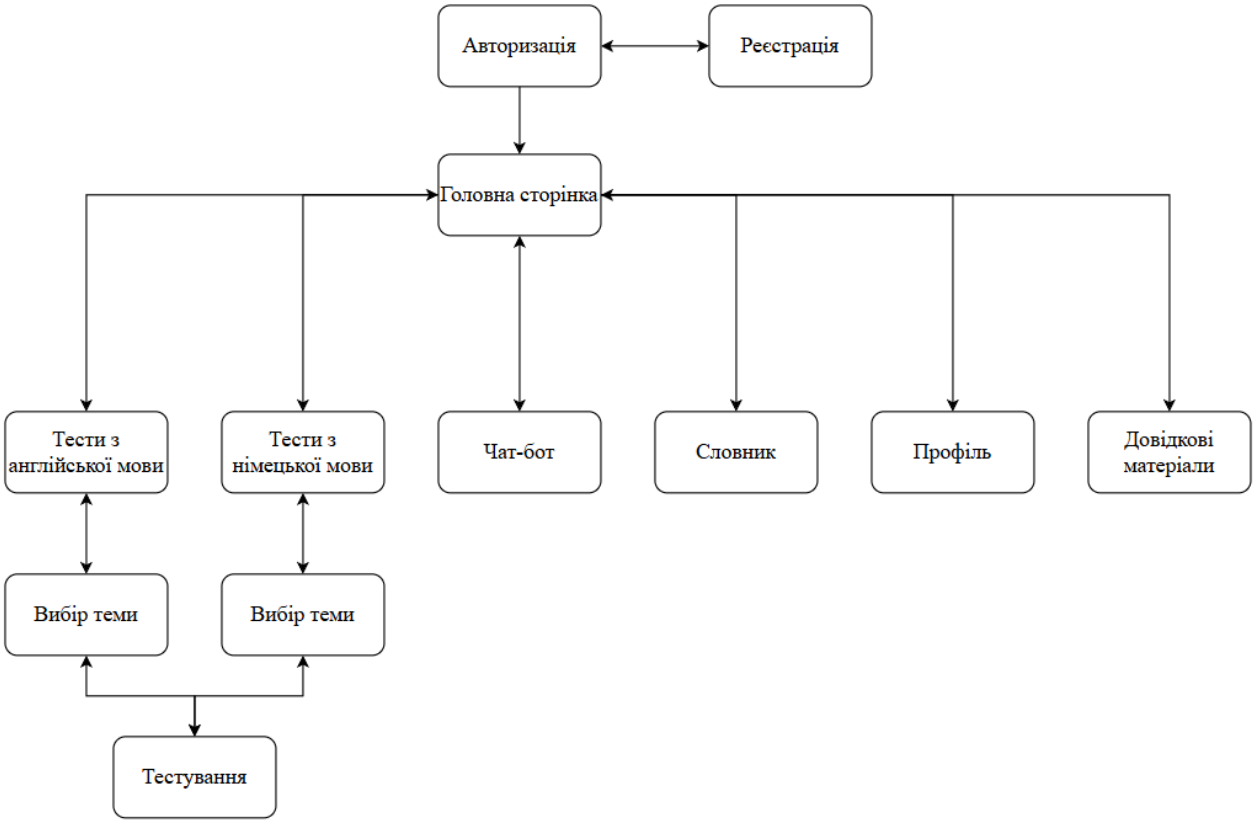


Рис. 3.2 Карта розроблюваного застосунку

3.3 Серверна частина

Серверна частина реалізована за допомогою сервісу Firebase та моделі штучного інтелекту llama3.2. Сервіс Firebase забезпечує масштабовану базу даних та бекенд-сервіси. Та llama3.2 яка хоститься на локальному сервері для потреб застосунка.

Основні функції Firebase в застосунку:

— реєстрація та авторизація користувачів. Firebase надає зручні інструменти для безпечного управління користувачами, а також має підтримку багатьох постачальників автентифікації (наприклад: Google, Facebook, GitHub);

— надійний захист даних користувачів за допомогою методів шифрування таких як модифікований scrypt;

— nosql база даних, яка реалізована в сервісі Firebase, що відповідає за збереження даних у вигляді пари ключ-значення або документів. Вона

забезпечує гнучку, масштабовану структуру, швидкий обмін даними в реальному часі та просту інтеграцію з іншими інструментами цього сервісу.

Основні функції llama3.2 в застосунку:

- пояснення помилок користувачів;
- функції чат-бота для тренування переписки;
- функції словника для детального пояснення значень слова.

3.4 Взаємодія компонентів

Архітектура мобільного застосунку побудована з урахуванням інтеграції з хмарним сервісом Firebase та моделлю штучного інтелекту llama3.2:

- користувацький інтерфейс, який реалізований за допомогою кросплатформного фреймворку Flutter від компанії Google, який дозволяє створювати високопродуктивні застосунки для платформ Android, iOS та Web з єдиною кодовою базою. Інтерфейс побудований на базі віджетів, що забезпечує гнучкість та спрощує підтримку застосунку;
- реєстрація та автентифікація користувачів, яка реалізована за допомогою сервісу Firebase Authentication з підтримкою авторизації через електронну пошту/пароль та Google OAuth. Це дозволяє забезпечити безпечну та зручну реєстрацію/автентифікацію;
- збереження даних, яке було реалізовано за рахунок використання сервісу Firebase та інструменту в ньому Firestore, яка є масштабованою хмарною базою даних, що дозволяє в режимі реального часу зберігати й синхронізувати дані;
- інтеграція зі штучним інтелектом для надання пояснень до помилок користувачів, реалізації чат-бота та словника. Застосунок надсилає запити до серверної частини де розгорнута модель штучного інтелекту llama3.2. Взаємодія відбувається за допомогою HTTP-запитів, а відповідь відображається безпосередньо в інтерфейсі користувача.

Така архітектура забезпечує масштабованість та простоту розвитку проекту, поєднуючи мінімалістичний дизайн, зручну авторизацію, хмарне зберігання та можливості генеративного штучного інтелекту. На рисунку 3.3 наведено діаграму варіантів використання розробленого застосунку.

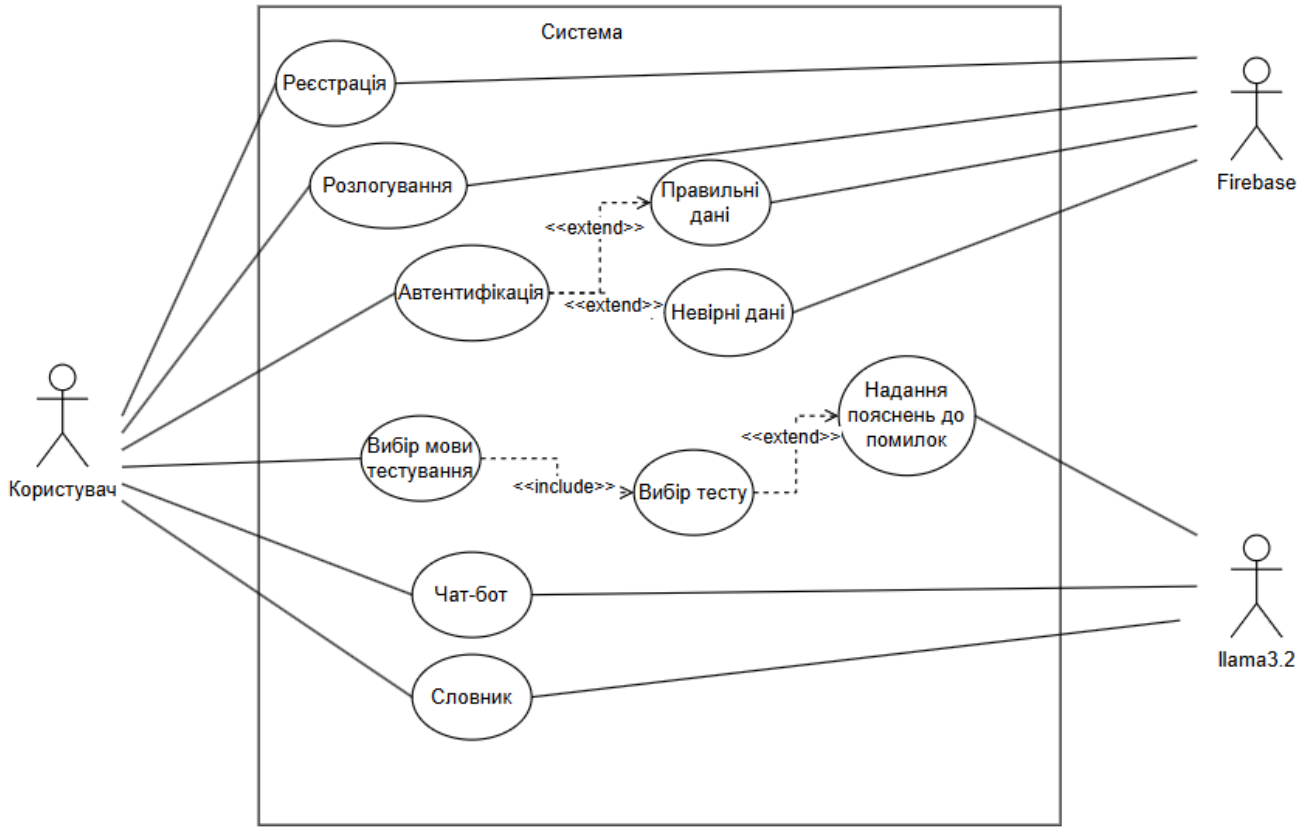


Рис. 3.3 Діаграма варіантів використання

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ЗАСТОСУНКУ

4.1 Дизайн інтерфейсу

Figma - це сучасний і потужний інструмент для створення інтерактивних прототипів інтерфейсу для мобільних застосунків, десктопних застосунків та веб сайтів. Нижче описано кроки виконані для створення прототипу UI за допомогою Figma.

1. Для створення нового проекту:
   - зареєструватися та увійти в Figma, виконавши наступні дії, відвідати сайт Figma, зареєструвати акаунт, якщо ще немає. Увійти в акаунт;
   - створити новий проект, за допомогою натискання на кнопку «New File» у панелі керування (Dashboard).
2. Створити фрейм:
   - додати фрейм натисканням на клавішу F або створити за допомогою інструменту Frame Tool;
   - вибрати фрейм зі списку шаблонів (наприклад iPhone, Desktop) або ж налаштувати його власноруч.
3. Додавання UI елементів на фрейм:
   - для створення базових елементів інтерфейсу використовуйте інструменти «Rectangle», «Line», «Ellipse»;
   - щоб додати текст натисніть на клавішу T або оберіть інструмент Text;
   - для вставки іконки та/або зображення імпортуйте їх зі свого комп’ютера чи скористайтеся вбудованими ресурсами Figma.
4. Прототипування:
   - перейдіть у вкладку Prototype;
   - виділіть бажаний об’єкт і протягніть від нього синю стрілку до фрейму на який має відбутися перехід.

— встановіть тип взаємодії, наприклад перехід на фрейм по натисканню чи відкриття оверлею;

— налаштуйте анімацію переходу у меню interaction details;

— для перевірки натисніть кнопку Present у верхньому правому куті й перегляньте інтерактивний макет. На рисунках 4.1, 4.2 показано екранні форми прототипів сторінок. На рисунку 4.3 показано загальний вигляд всіх розроблених прототипів.

Рис. 4.1 Екранна форма створеного прототипу головної сторінки в Figma

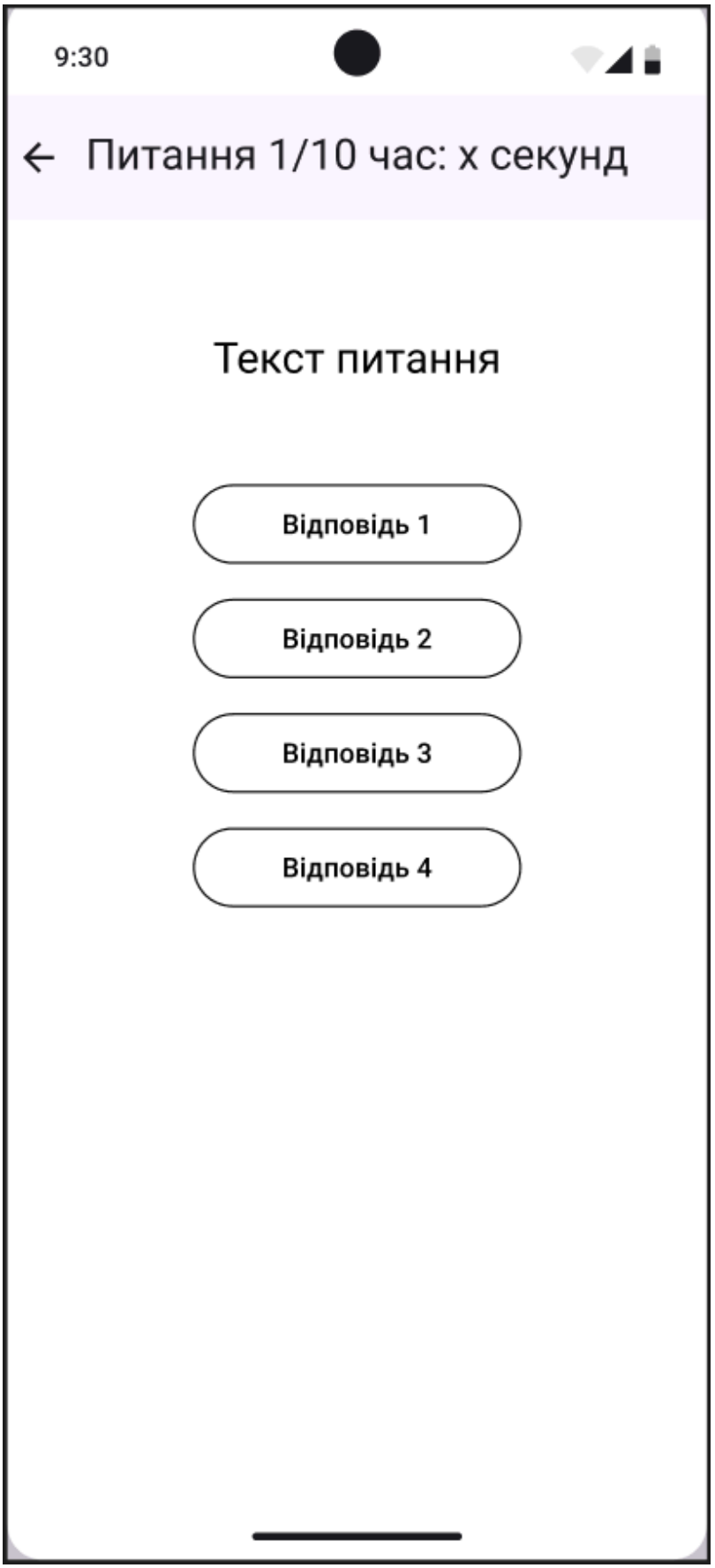


Рис. 4.2 Екранна форма створеного прототипу сторінки з тестами в Figma

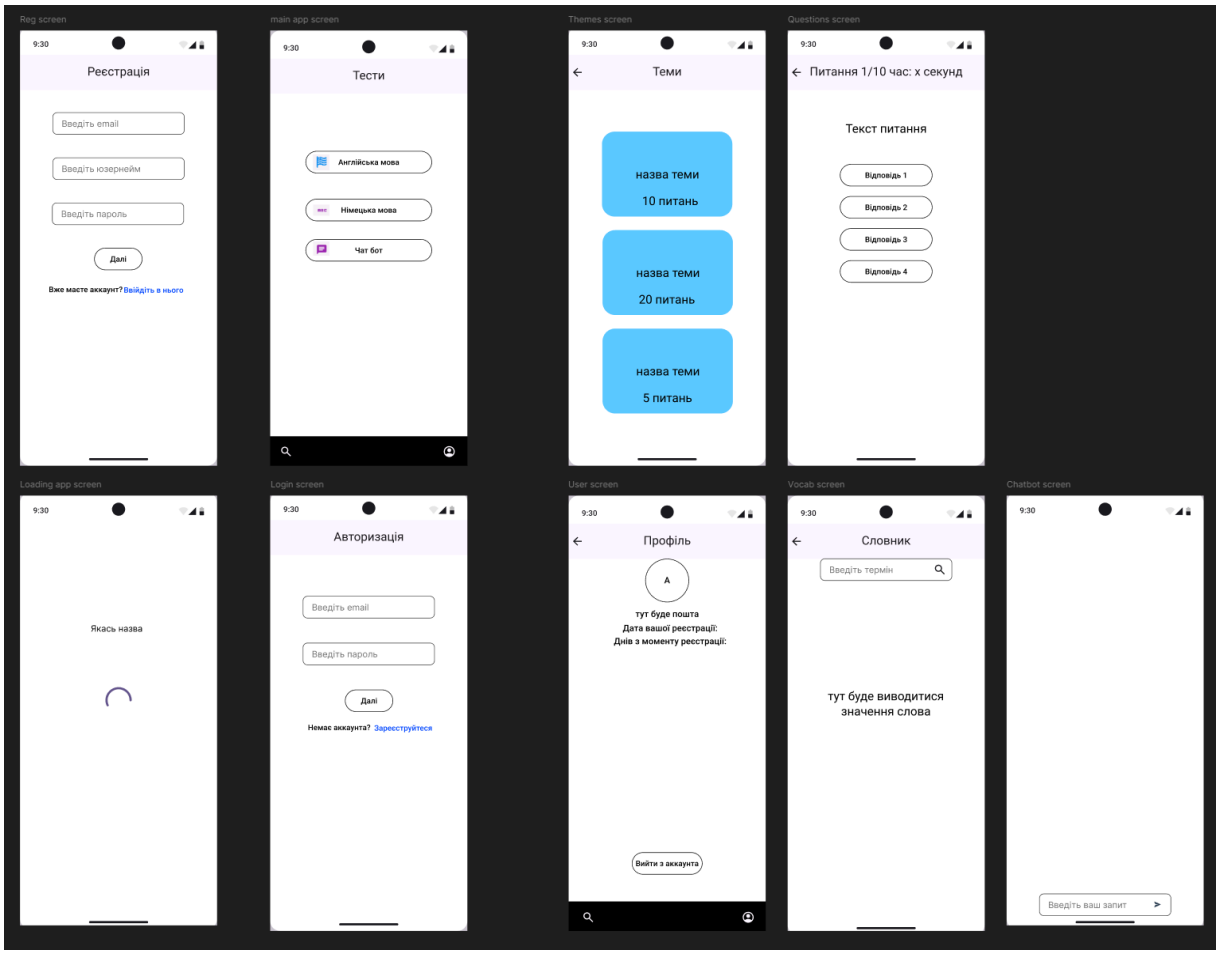


Рис. 4.3 Загальний вид прототипів всіх сторінок

4.2 Екран авторизації

Екран авторизації - це екран з якого користувач починає взаємодіяти з застосунком. Вхід до застосунку відбувається за допомогою електронної пошти та паролю або через Google акаунт користувача. Якщо ж у користувача немає акаунта він з легкістю може перейти на екран реєстрації за допомогою кнопки «Зареєструйтеся». На рисунку 4.4 показано розроблений екран авторизації в мобільному застосунку.

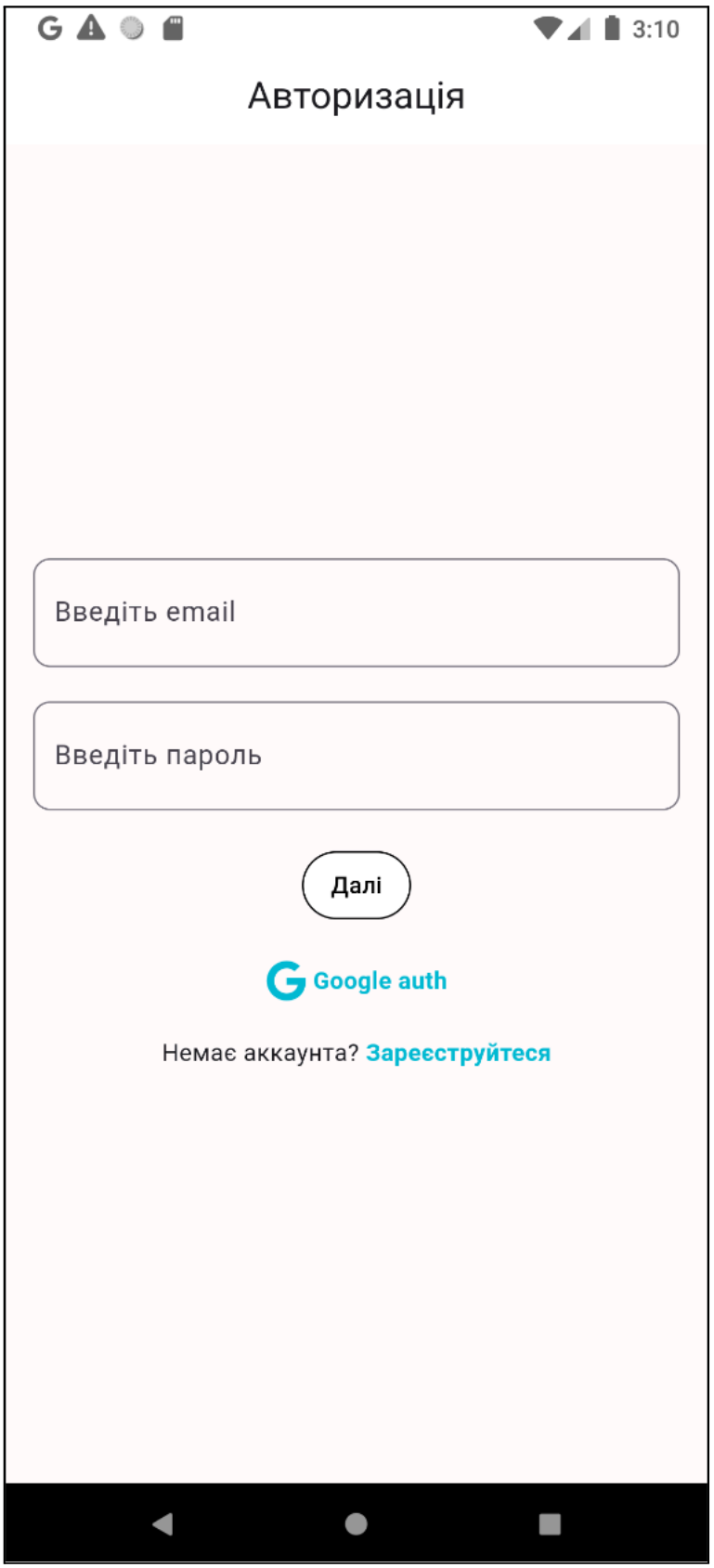


Рис. 4.4 Екран авторизації в застосунку

На рисунку 4.5 показано, реагує розроблений мобільний застосунок реагує на введення неправильних даних при спробі увійти в акаунт.

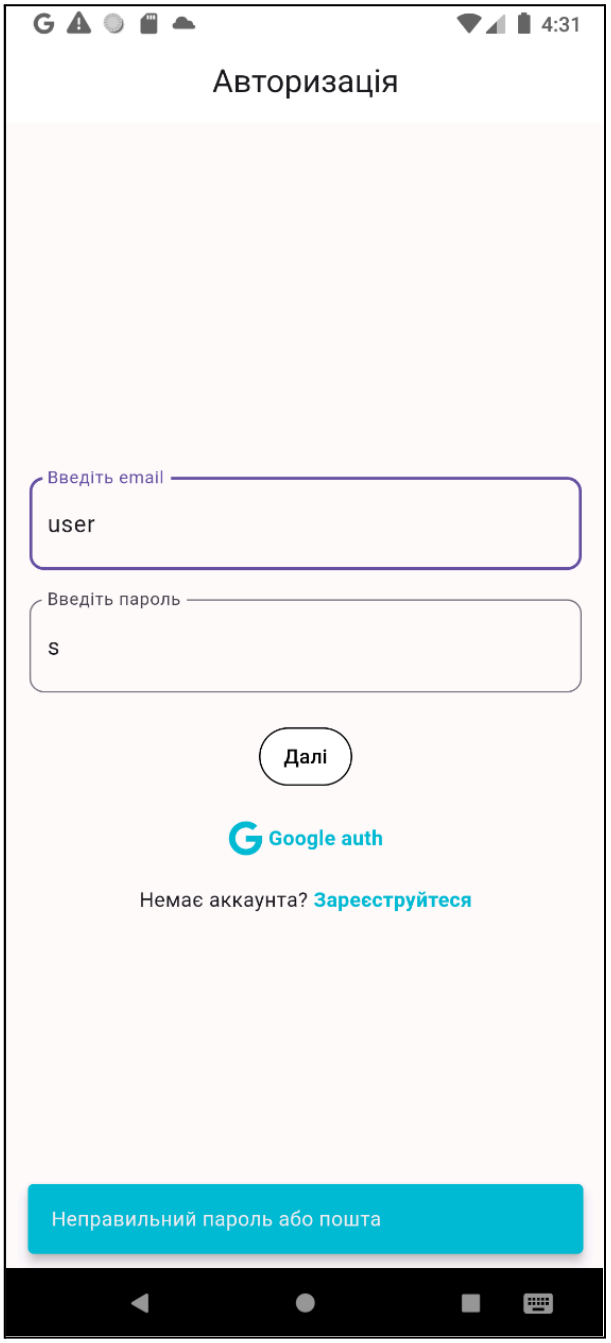


Рис. 4.5 Інформаційне повідомлення при вводі неправильних даних при вході в акаунт

При спробі користувача увійти в неіснуючий акаунт чи з неправильними даними він отримає інформаційне повідомлення з помилкою. Відображення інформаційного повідомлення реалізовано за допомогою вбудованого в Flutter методу SnackBar().

4.3 Екран реєстрації

Для реєстрації у застосунку користувачу потрібно ввести ім’я користувача, електронну пошту та пароль. На рисунку 4.6 показано розроблений екран реєстрації в мобільному застосунку.

Рис. 4.6 Екран реєстрації в застосунку

На рисунку 4.7 показано, як реагує розроблений мобільний застосунок на неправильні дані під час реєстрації.

Рис. 4.7 Інформаційне повідомлення при введенні неправильних даних при реєстрації

Як і при вході під час реєстрації відбувається перевірка на валідність введених даних. Якщо користувач вводить електронну пошту в неправильному

форматі він отримає інформаційне повідомлення. Також відбувається перевірка на правильність формату пароля.

На рисунку 4.8 показано, як реагує розроблений мобільний застосунок на спробу зареєструвати акаунт вже існуючий акаунт.

Рис. 4.8 Спроба зареєструвати вже існуючий акаунт

При спробі створити акаунт з невірними даними або вже існуючими в Firebase, користувач отримає інформаційне повідомлення з відповідною

помилкою. Весь цей функціонал реалізований в файлі «firebase_auth_service» за допомогою використання бібліотеки «firebase_auth».

4.4 Головний екран

Головний екран - це екран з якого користувач може взаємодіяти з основним функціоналом застосунку.

Рис. 4.9 Головний екран

На рисунку 4.9 зображений головний екран, він виконаний у світлому простому та зручному форматі. Які є максимально зрозумілими та зручними у використанні. Інтерфейс побудований на базі віджетів, які є основою для будування користувацького інтерфейсу в Flutter. Віджети у Flutter можуть бути як простими кнопками, так і складніші компоненти, форми чи списки. Основним віджетом, що формує структуру всього застосунку MaterialApp. За допомогою нього в застосунку реалізується Material Design розроблена компанією Google. Будівельним каркасом слугує віджет Scaffold, який містить у собі AppBar, Body, BottomNavigationBar. AppBar реалізує шапку в застосунку. Body реалізує основний вміст сторінки, який складається з трьох кнопок: «Англійська мова», «Німецька мова» та «Чат-бот». Натискання на кожну з цих кнопок ініціює перехід на відповідну сторінку за допомогою механізму навігації Navigator. BottomNavigationBar реалізує нижню навігаційну панель, яка надає доступ до наступних розділів: «Словник», «Теорія» та «Профіль».

4.5 Екран вибору теми

Екран вибору теми - це екран на якому користувач може обрати тему для проходження тестування. Всього в застосунку таких екрана два. Один для англійської мови та інший для німецької. Обидва цих екрани реалізовані за допомогою патерна MVC та принципу повторного використання коду. На рисунку 4.10 показано розроблений екран вибору теми для англійської мови.

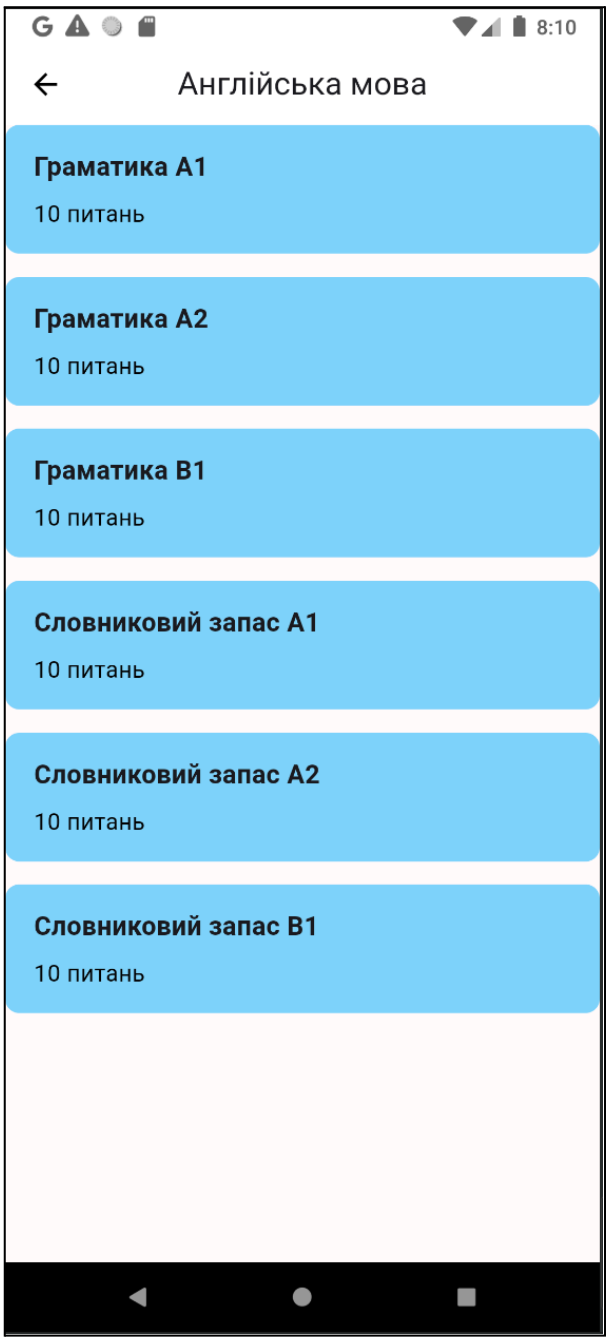


Рис. 4.10 Екран вибору теми в англійській мові

Екран з вибором тем з англійської мови реалізується компонентом «EnglishQuestions», який є StatelessWidget. Він відповідає за відображення екрану зі списком тем для англійської мови, цей компонент викликає віджет «TopicListScreen» передаючи йому назву мови, як параметр. Цей віджет в свою чергу описує загальний вигляд екрану і викликає конструктор «TopicController» з передачею назви мови. На основі цього створюється екземпляр «TopicController» з відповідною мовою і повертається список тем, який потім

динамічно рендериться за допомогою віджета «ListView.builder». Це дозволяє використовувати один і той самий віджет для різних мов, передаючи йому відповідну конфігурацію під час ініціалізації. Така реалізація також дозволяє з легкістю масштабувати застосунок, як от додати нову тему чи нову мову для тестування. На рисунку 4.11 показано структурну діаграму роботи екрану з вибором тем.

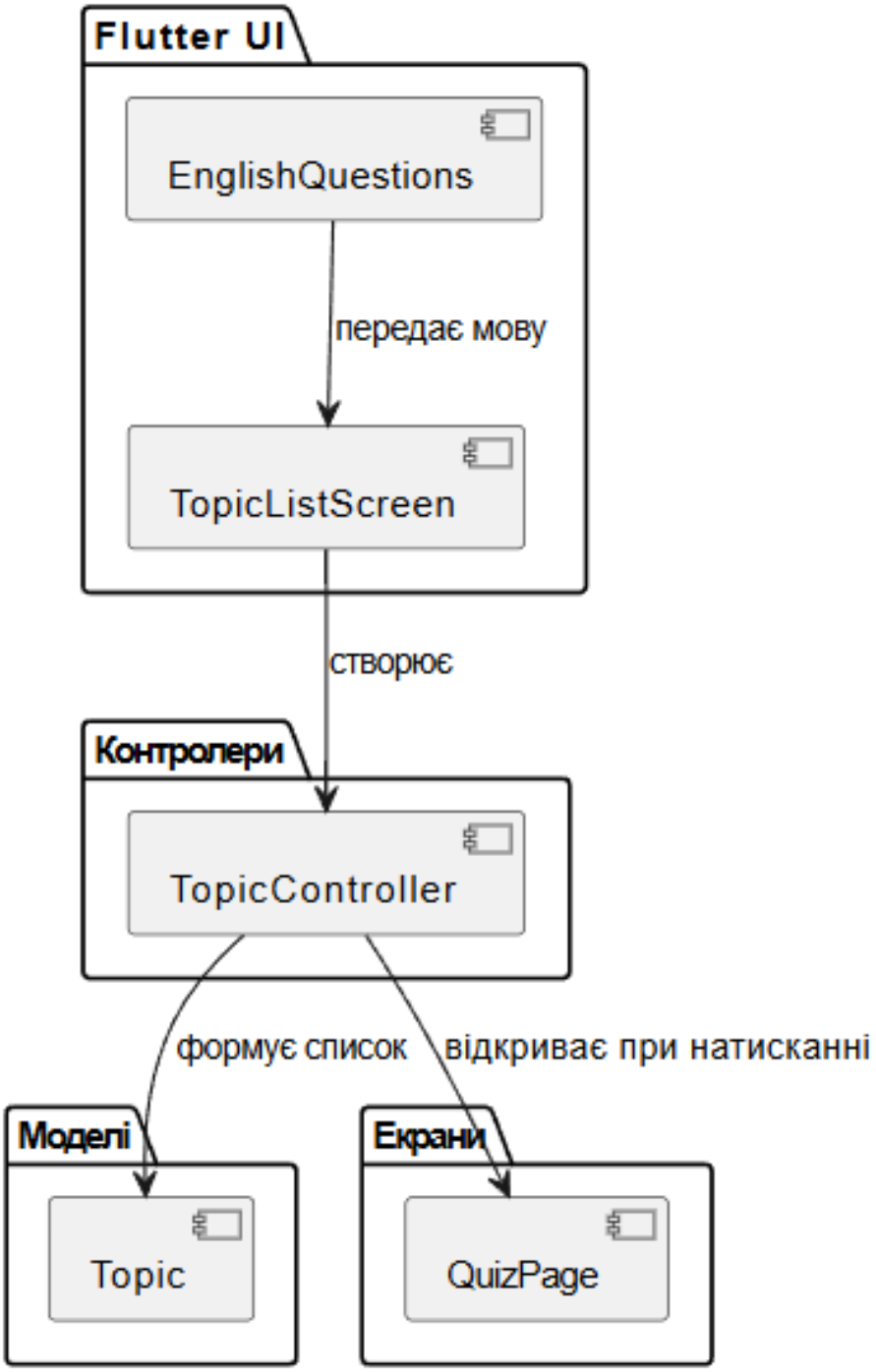


Рис. 4.11 Структурна діаграма роботи екрану вибору теми

На рисунку 4.12 показано розроблений екран вибору тем для німецької мови.

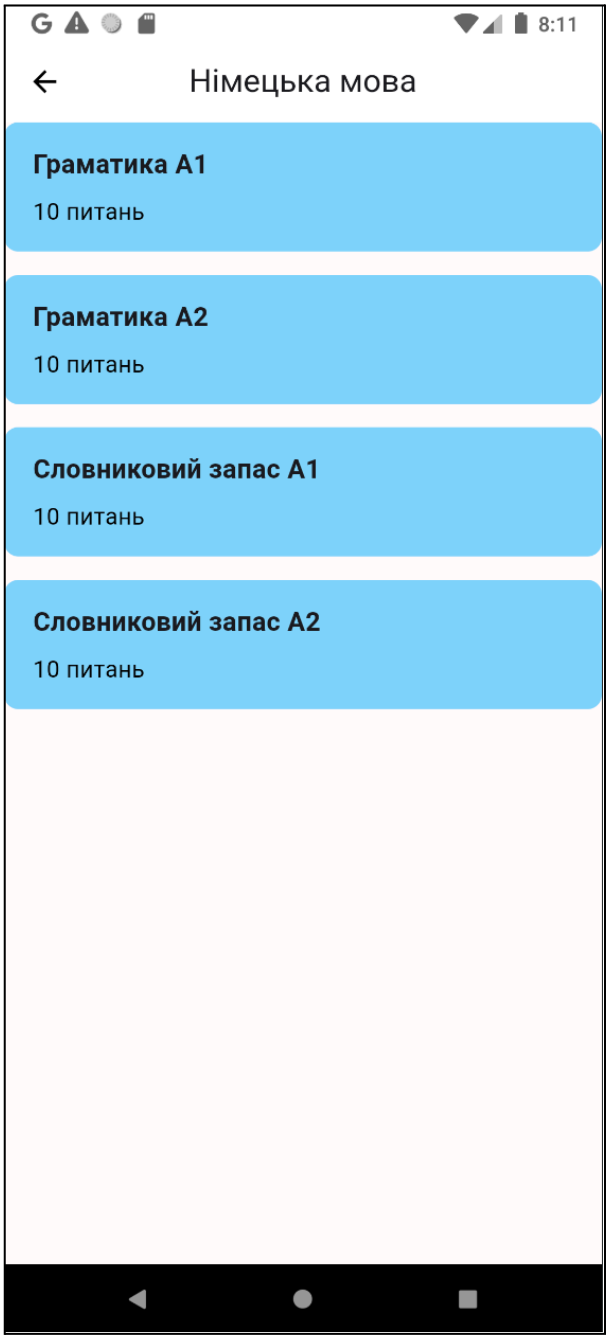


Рис. 4.12 Екран вибору теми в німецькій мові

Екран з вибором тем з німецької мови реалізований точно так же, як і відповідний йому екран з вибором тем з англійської мови. Всього на даний момент питань з цих двох мов 100 і вони розбиті на 10 тем по складності і категоріях.

4.6 Екран тестування

Екран тестування - це екран на якому користувач послідовно відповідає на питання з обраної теми на екрані вибору теми. На рисунку 4.13 показано розроблений екран тестування.

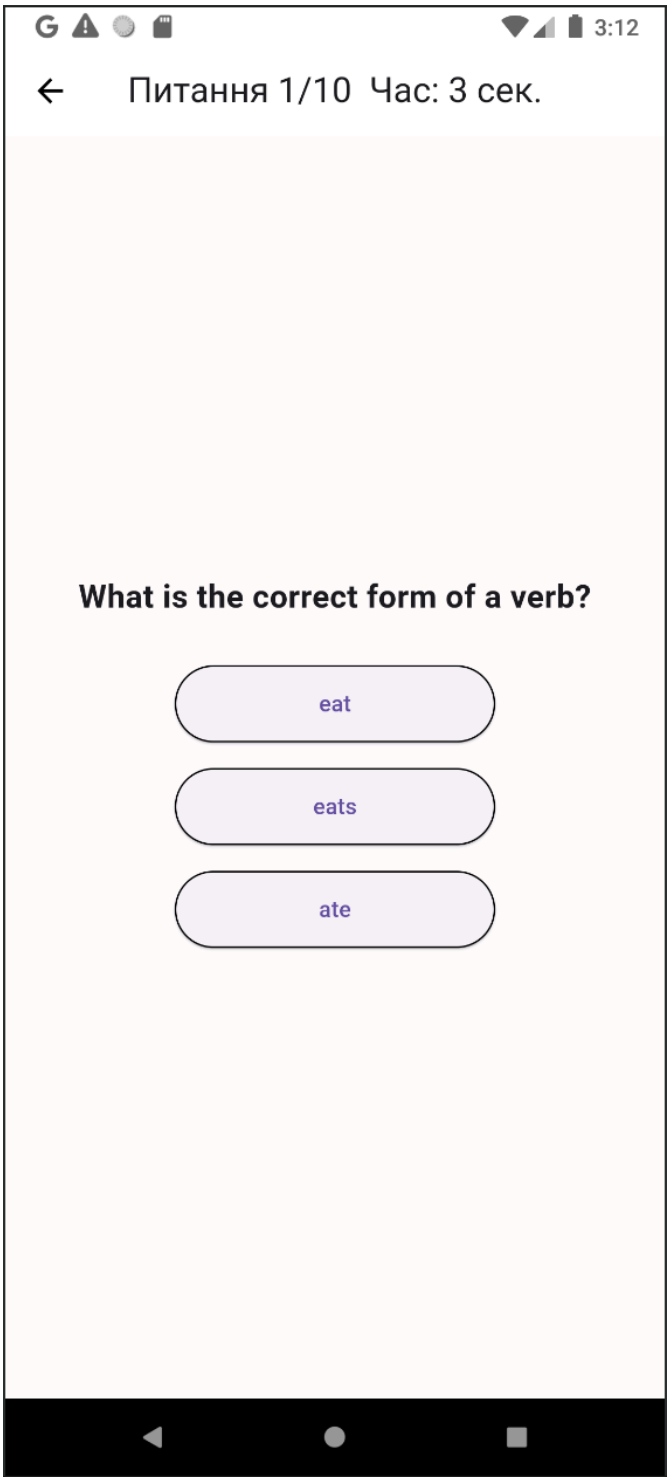


Рис. 4.13 Екран тестування

Він реалізований з використанням патерну MVC з чітким розділенням логіки, моделі та відображення. Відображення представляє клас «QuizPage», який є «StatefulWidget» і відповідає за динамічне оновлення сторінки під час проходження тесту. За логіку відповідає контроллер «QuizController», який завантажує список питань з Firebase Firestore на основі «topicId», переданого з екрану вибору теми. Після завантаження починається таймер, який щосекунди оновлює інтерфейс, відображаючи час витрачений на тестування. Після завершення тестування таймер зупиняється, а результати зберігаються в Firestore з наступними даними: результатом, часом затраченим на тестування у секундах, датою тестування, темою тестування, кількістю питань в темі та uid користувача в Firebase. Після цього користувач направляється на екран з результатами.

4.7 Вікно з результатами

Вікно з результатами - це вікно на якому користувач бачить свої результати, час витрачений на тест та має можливість отримати розбір помилок від моделі штучного інтелекту llama3.2. При натисканні на кнопку «Розбір помилок» відкривається вікно з аналізом помилок. На рисунку 4.14 показано спливаюче вікно з результатами, яке з’являється після проходження тесту.

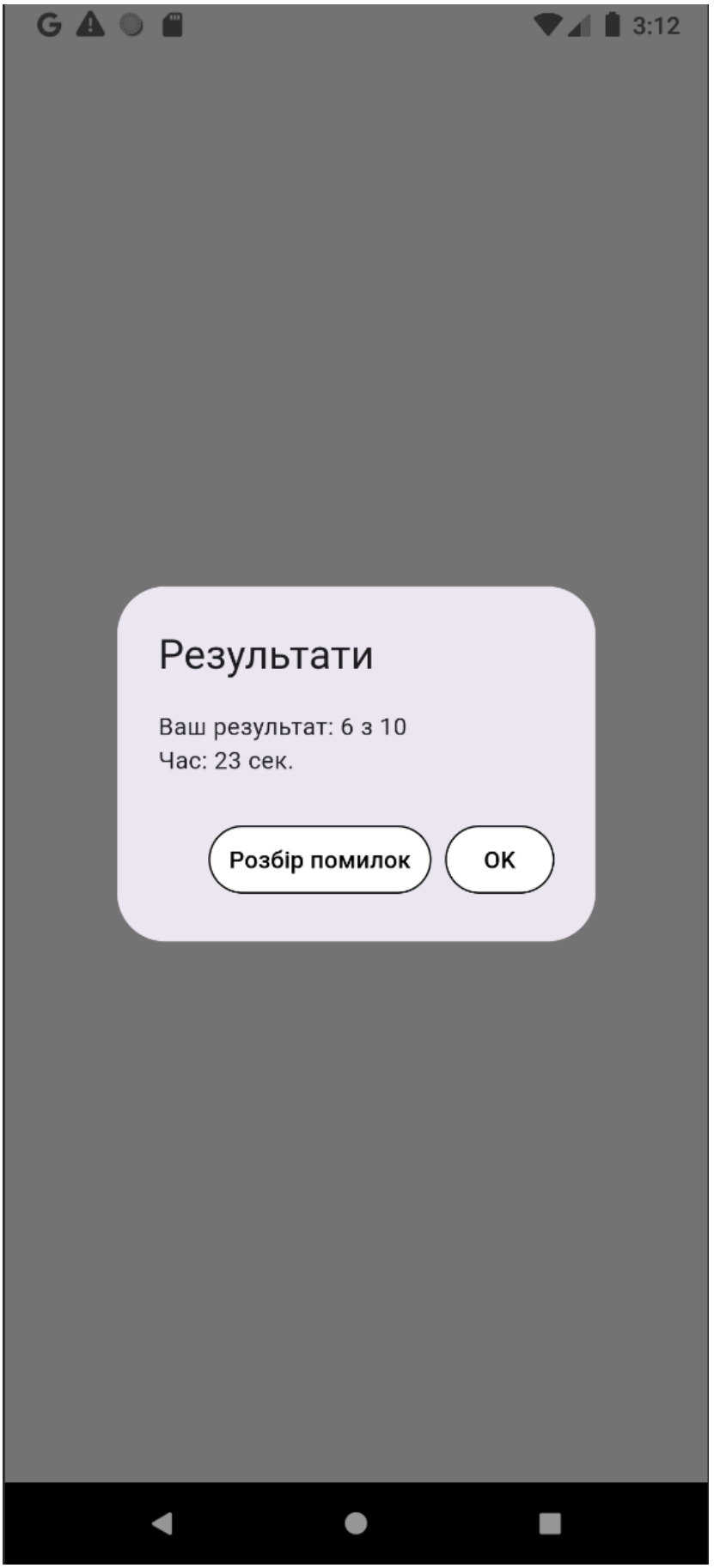


Рис. 4.14 Вікно з результатами

4.8 Вікно розбору помилок

Після того, як користувач натискає на кнопку «Розбір помилок» у вікні результати тестування його відповіді опрацьовуються за наступним алгоритмом:

1. Закриття вікна результатів.

Поточне діалогове вікно з результатами тесту закривається для переходу до етапу аналізу помилок.

2. Відображення індикатора завантаження.

Користувачеві показується нове діалогове вікно із заголовком «Аналіз помилок» та індикатором завантаження, що свідчить про обробку даних.

3. Формування та надсилання запиту на аналіз.

Внутрішній контролер системи здійснює обробку списку помилок, виявлених у процесі тестування. Для кожного запитання, на яке було надано неправильну відповідь, формується інформація, що включає:

- текст запитання;
- обрану користувачем відповідь;
- правильну відповідь.

Зазначені дані об'єднуються у структурований запит, який надсилається до локально запущеної моделі ШІ (LLM) для генерації пояснень.

4. Отримання та обробка відповіді.

Після отримання відповіді від моделі ШІ llama3.2, що містить пояснення кожної помилки, індикатор завантаження закривається.

5. Відображення аналізу помилок.

Користувачеві демонструється нове діалогове вікно із заголовком «Розбір помилок», в якому покроково подано:

- питання, в якому була допущена помилка;
- варіант, обраний користувачем;
- правильну відповідь;
- пояснення помилки, сформульоване мовною моделлю.

Інформація надається у зручному для перегляду форматі, з можливістю прокручування.

6. Завершення процесу.

Після ознайомлення з розбором користувач може закрити вікно натисканням кнопки підтвердження (наприклад, «OK»), що повертає його на головну сторінку застосунку або до попереднього екрану.

Такий механізм спрямований на поглиблення розуміння навчального матеріалу користувачем, дозволяє ефективно проаналізувати помилки та сприяє підвищенню якості засвоєння інформації. На рисунку 4.15 показано розроблене спливаюче вікно з розбором помилок користувача в тесті від ШІ.

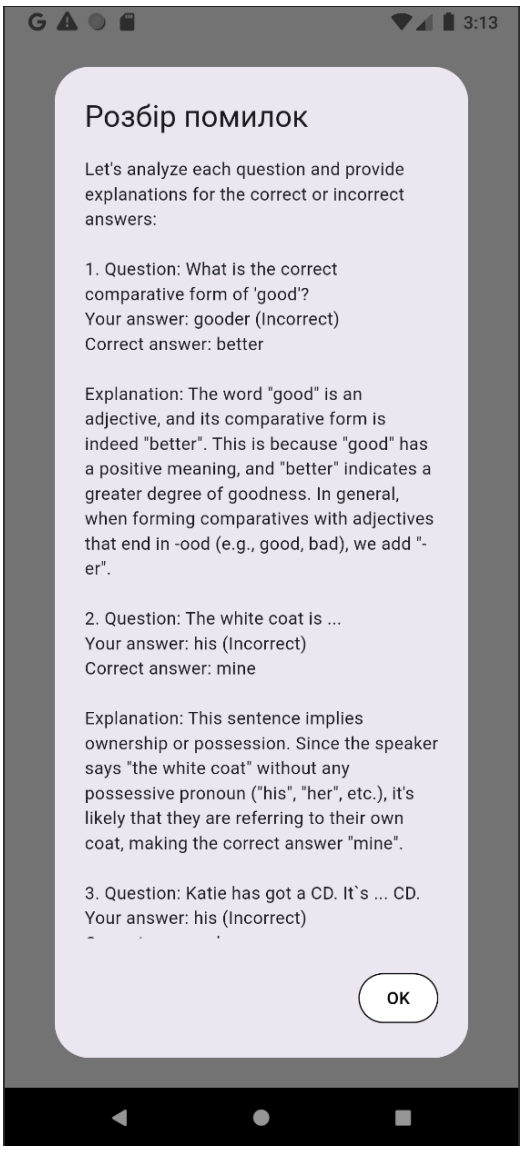


Рис. 4.15 Вікно розбору помилок

4.9 Екран чат-боту

Екран чат-боту - це екран в якому користувач може попрактикуватися в використанні мови з чат-ботом. Під час спілкування користувач має змогу обрати бажану тему розмови, а також буде отримувати поради від моделі штучного інтелекту поради з покращення своїх навичок в практичному використанні мови. На рисунку 4.16 показано розроблений екран чат-боту.

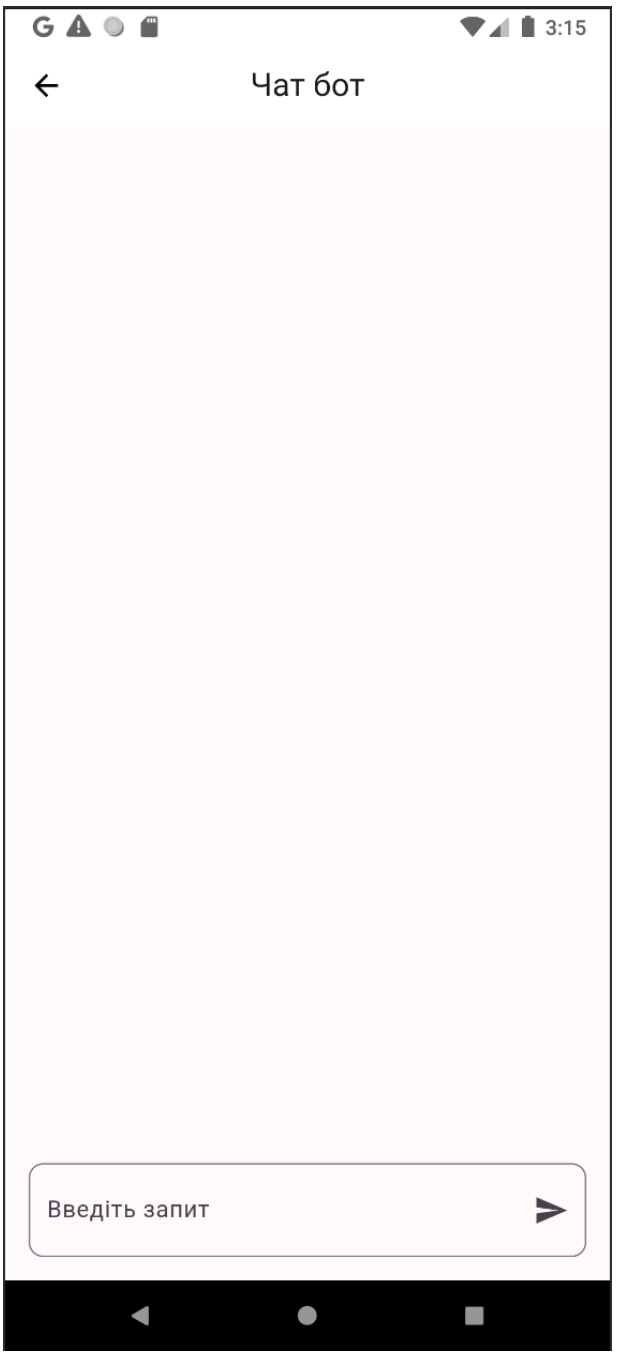


Рис. 4.16 Екран чат-боту

Цей екран працює наступним чином:

1. Ініціалізація

Після завантаження екрану створюється підписка на потік повідомлень із моделі. Це забезпечує автоматичне оновлення інтерфейсу при кожній зміні в чаті.

2. Відображення повідомлень

Всі повідомлення відображаються у вигляді прокручуваного списку. Кожне повідомлення виводиться в окремому контейнері, колір якого залежить від ролі: у користувача повідомлення зеленого кольору, відповіді штучного інтелекту синього кольору.

3. Ввід тексту

У нижній частині екрана розташоване текстове поле, яке дозволяє користувачеві вводити запити.

4. Обробка натискання кнопки надсилання

Якщо текстове поле не порожнє, введене повідомлення передається до контролера для подальшої обробки, а саме текстове поле очищується.

4.10 Екран словника

Екран словника - це екран в якому користувач може дізнатися значення бажаного терміну та як його використовувати на практиці. Також підтримується голосовий ввід та можливість зачитки відповіді від моделі штучного інтелекту.

Архітектура побудована за принципами розділення відповідальностей (MVC), що забезпечує зручність у підтримці та масштабуванні коду. На рисунку 4.17 показано розроблений екран словника.

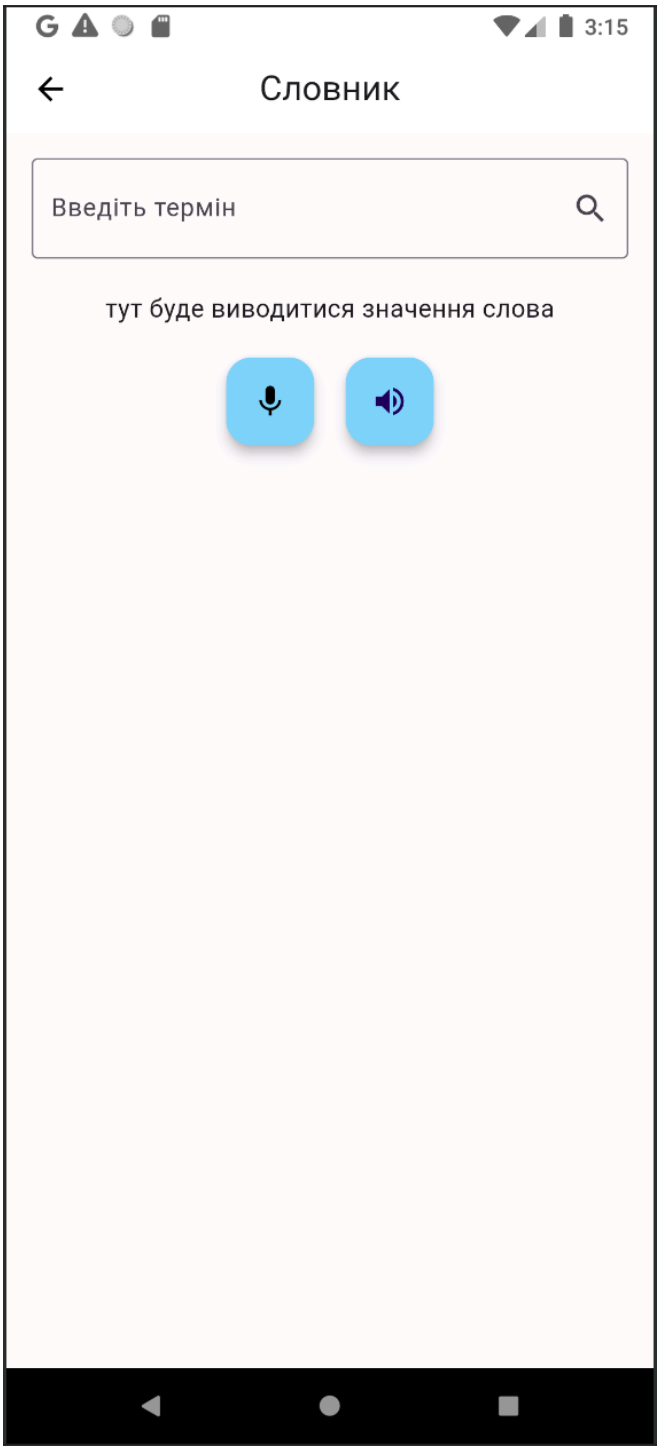


Рис. 4.17 Екран словника

Загальний опис функціональності:

Графічний інтерфейс «SearchScreen» представляє користувацький інтерфейс для взаємодії з додатком. Складається з:

- поля введення тексту для введення терміна;
- кнопки для ініціювання пошуку;

— виводу результату у вигляді визначення слова;

— голосових кнопок для розпізнавання мови представлений іконкою мікрофону та озвучування тексту представлене гучномовцем.

Контролер «SearchWordController»

Відповідає за логіку зв’язку між UI та моделлю. Контролює стан елементів інтерфейсу за допомогою «ValueNotifier» і обробляє події натискання кнопок. Синхронізує відображення результатів пошуку, статус завантаження, розпізнавання та озвучування.

Модель «SearchModel» забезпечує бізнес-логіку:

— використовує API штучного інтелекту для отримання пояснення введеного терміна;

— реалізує голосове введення за допомогою бібліотеки speech_to_text;

— реалізує озвучування результату через flutter_tts;

— управляє внутрішнім станом (результат, стан мови, мікрофону, завантаження тощо).

Основні можливості:

— пошук значення терміна з використанням моделі ШІ llama3.2;

— голосове введення терміна;

— озвучування знайденого значення;

— динамічне оновлення інтерфейсу відповідно до стану програми.

Цей підхід дозволяє користувачу здійснювати пошук термінів як вручну, так і голосом, а також отримувати результат у вигляді озвученого тексту, що покращує доступність та зручність взаємодії з додатком.

4.11 Екран профілю

Екран профілю має простий та зрозумілий дизайн. На ньому знаходиться основна інформація про користувача така як: кількість днів з реєстрації, середній бал по тестах. Та можливість вийти з акаунта. На рисунку 4.18 показано екран профілю.

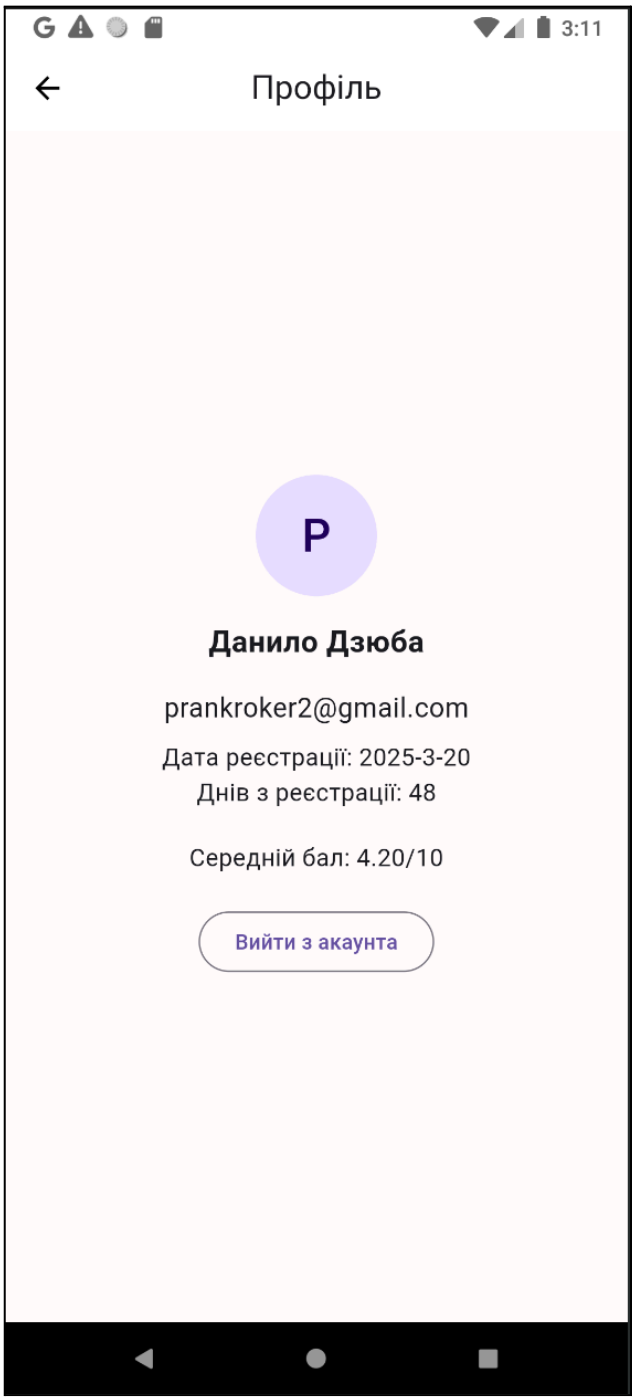


Рис. 4.18 Екран профілю

Угорі екрану знаходиться заголовок екрану, і як і на інших екранах це реалізовано за допомогою вбудованого в Flutter віджету AppBar. Далі йде аватар користувача, поле з ім’ям, електронна пошта, дата реєстрації, кількість днів з реєстрації та середній бал по тестам. І останній елемент кнопка виходу з акаунта.

4.12 Екран довідкових матеріалів

Екран довідкових матеріалів - це екран на якому знаходяться посилання на довідкові матеріали з англійської та німецької мов. На рисунку 4.19 показано екран довідкових матеріалів.

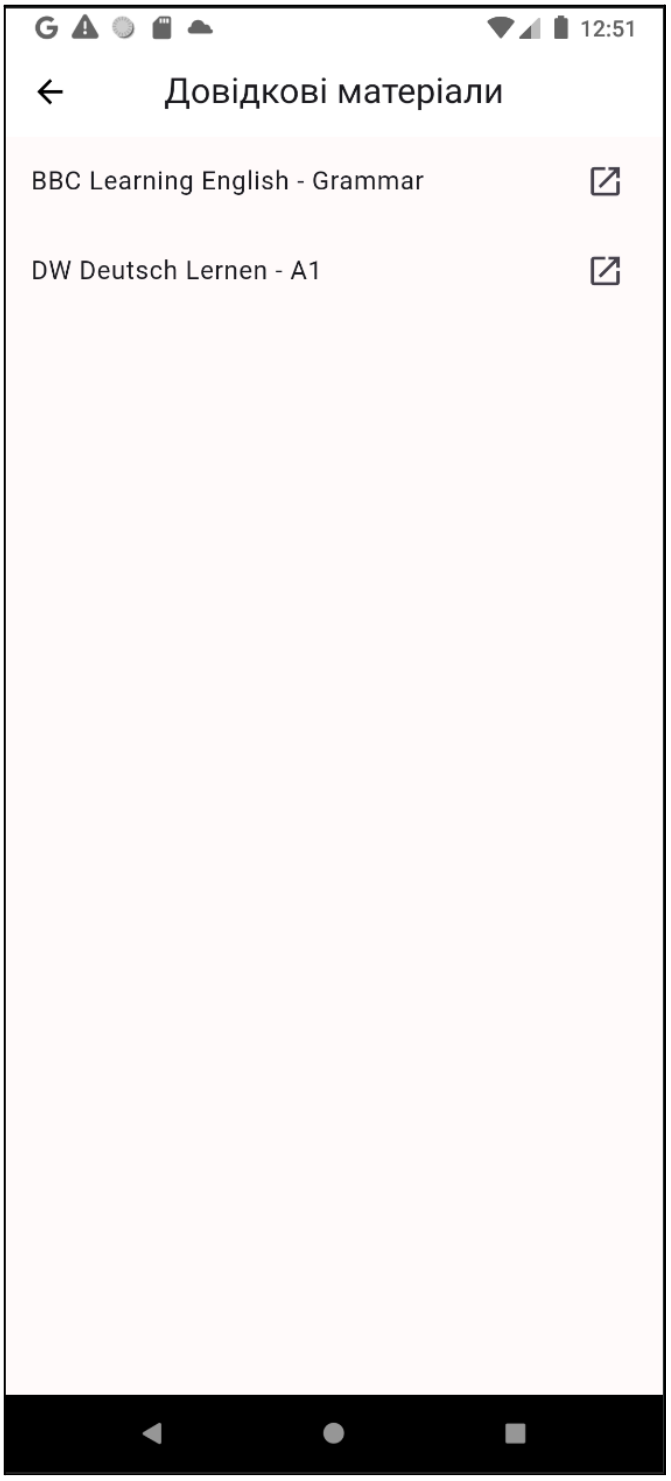


Рис. 4.19 Екран довідкових матеріалів

Перехід на цей екран відбувається з головного екрану (пункт 4.4) натисканням на кнопку «Теорія», за допомогою класу Navigator вбудованого в фреймворк Flutter.

4.13 Тестування застосунку

Тестування це критично важливий етап в розробці будь-якого застосунку, адже тестування дозволяє забезпечити його надійність, продуктивність та відповідність поставленим вимогам.

У цьому розділі буде описано основні принципи та методи тестування, які використовуються в фреймворці Flutter. Також буде надано приклад тестів для мобільного застосунку з тестами з іноземної мови з використанням технологій штучного інтелекту.

Тестування програмного забезпечення у Flutter містить в собі 3 основних методи, а саме юніт тести, віджет тести та інтеграційні тести.

Юніт тести

Юніт тести призначені для тестування однієї окремої функції, методу або класу. Їх задача полягає в перевірці правильності роботи певної функції, методу чи класу. Зовнішні залежності в таких тестах зазвичай передаються як параметр. Основними характеристиками юніт тестів є ізольованість, автоматизація, незалежність та повторюваність.

Для написання юніт тестів у Flutter використовують вбудований пакет test. Він дозволяє створювати та виконувати тести застосунку.

Віджет тести

Віджет тести призначені для тестування одного віджета. Задача такого тесту впевнитися, що користувацький інтерфейс віджета виглядає та працює так як було заплановано. Тестування віджета проводиться в тестовому середовищі, яке забезпечує контекст життєвого циклу віджета. Також віджет який тестують повинен мати змогу отримувати дії від користувача і відповідати на них.

Для написання віджет тестів у Flutter, як і для юніт тестів використовується вбудований пакет test. Він дозволяє створювати та виконувати тести застосунку.

Інтеграційні тести

Інтеграційні тести перевіряють чи правильно взаємодіють між собою різні модулі та компоненти застосунку. Задача такого тесту впевнитися, що всі віджети, модулі, компоненти та сервіси, які тестують працюють разом як і очікувалось. Крім того, інтеграційні тести можна використовувати для перевірки продуктивності розробленого застосунку. Все через те що інтеграційні тести виконуються на реальному пристрої чи емуляторі.

Для написання інтеграційних тестів у Flutter використовують пакет integration_test. Цей пакет дозволяє створювати та виконувати інтеграційні тести застосунку.

Нижче наведено таблицю 4.1 компромісів під час тестування:

Таблиця 4.1

Таблиця компромісів під час проведення тестування

Критерій	**Юніт тест**	**Віджет тест**	**Інтеграційний тест**
Впевненість в застосунку після тесту	Низька	Середня	Висока
Складність реалізації	Легка	Складно	Дуже складно
Залежності	Майже відсутні	Доволі багато	Дуже багато
Час виконання	Швидко	Швидко	Повільно

З цієї таблиці можна побачити, що юніт тести дають нам низьку впевненість в застосунку, але при цьому вони реалізуються відносно легко та

швидко виконуються. Віджет тести є компромісом між впевненістю в застосунку, складніст. реалізації та часом виконання. Інтеграційні ж тести надають найбільшу впевненість в правильності роботи застосунку, але їх реалізація та виконання займає значно більше часу.

Нижче на рисунку 4.20 показано приклад віджет тесту в розробленому мобільному застосунку.

```dart
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';
import 'package:flutter_diploma/views/screens/mainPage.dart';

void main() {
  testWidgets('На головному екрані мають бути присутні всі UI елементи', (WidgetTester tester) async {
    await tester.pumpWidget(
      const MaterialApp(
      home: MainPage(),
      ),  // MaterialApp
    );
    expect(find.text('Англійська мова'), findsOneWidget);
    expect(find.text('Німецька мова'), findsOneWidget);
    expect(find.text('Чат-бот'),findsOneWidget);

    expect(find.byIcon(Icons.search), findsOneWidget);
    expect(find.byIcon(Icons.book), findsOneWidget);
    expect(find.byIcon(Icons.person), findsOneWidget);
  });
}
```

Рис. 4.20 Приклад коду для віджет тесту

Нижче на рисунку 4.21 показано результат проходження тесту з рисунку 4.13.1.

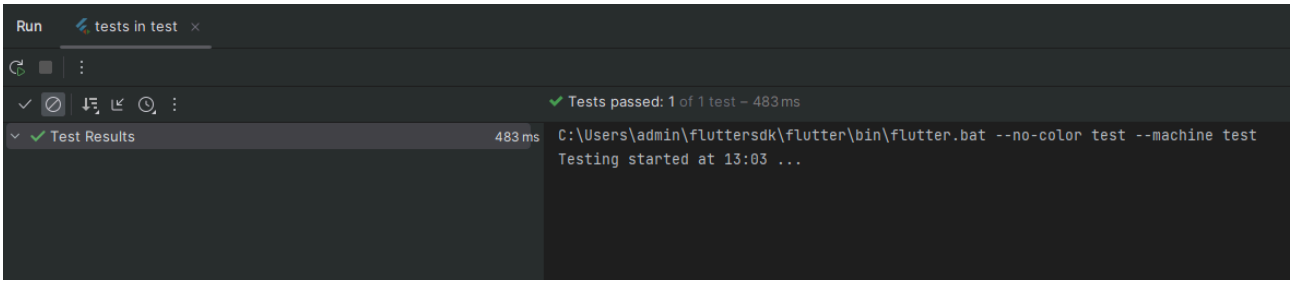


Рис. 4.21 Результат виконання віджет тесту

Цей простий віджет тест перевіряє чи присутні всі необхідні UI елементи на головному екрані після переходу на нього.

Нижче на рисунку 4.22 наведено приклад інтеграційного тесту в розробленому мобільному застосунку.

```
void main() {
  IntegrationTestWidgetsFlutterBinding.ensureInitialized();
  group('Тестування автентифікації', () {
    testWidgets('Невірні дані', (WidgetTester tester) async {
      app.main();
      await tester.pumpAndSettle();
      await Future.delayed(const Duration(seconds: 2));
      await tester.pumpAndSettle();
      expect(find.byType(Login), findsOneWidget);
      final emailField = find.byKey(const ValueKey('email_field'));
      final passwordField = find.byKey(const ValueKey('password_field'));
      await tester.enterText(emailField, 'test@example.com');
      await tester.enterText(passwordField, 'password123');
      await tester.tap(find.text('Далі'));
      await tester.pumpAndSettle();
      await Future.delayed(const Duration(seconds: 1));
      expect(find.text('Неправильний пароль або пошта'), findsOneWidget);
    });
    testWidgets('Вірні дані', (WidgetTester tester) async {
      app.main();
      await tester.pumpAndSettle();
      await Future.delayed(const Duration(seconds: 2));
      await tester.pumpAndSettle();
      expect(find.byType(Login), findsOneWidget);
      final emailField = find.byKey(const ValueKey('email_field'));
      final passwordField = find.byKey(const ValueKey('password_field'));
      await tester.enterText(emailField, 'tester1@gmail.com');
      await tester.enterText(passwordField, 'tester1');
      await tester.tap(find.text('Далі'));
      await tester.pumpAndSettle();
      await Future.delayed(const Duration(seconds: 1));
      expect(find.byType(MainPage), findsOneWidget);
    });
  });
```

Рис. 4.22 Приклад коду інтеграційного тесту

Нижче на рисунку 4.23 показано результат проходження інтеграційних тестів.

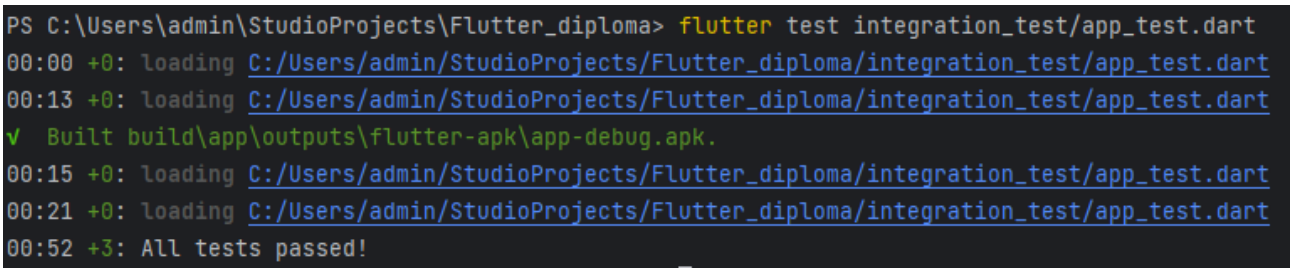


Рис. 4.23 Результат виконання інтеграційних тестів

Цей інтеграційний тест дозволяє на реальному емуляторі мобільного пристрою з системою Android, перевірити правильність роботи автентифікації в застосунку. Перший тест проводиться на невірних даних для перевірки працездатності системи з неправильними вхідними даними. Другий тест проводиться з вірними даними і слугує для перевірки працездатності системи на вірних даних і автоматичного переходу користувача на головну сторінку застосунку після автентифікації.

ВИСНОВКИ

Під час роботи над кваліфікаційною роботою отримано результати::

1.Проаналізовано предметну галузь, у результаті чого підтверджено актуальність розробки мобільного застосунку та встановлено, що існує сталий попит на інструменти для самонавчання з використання штучного інтелекту.

2.Проведено аналіз аналогів, а саме Duolingo, Lingvist, Babbel. Виявлено їх обмеження, які будуть конкурентними перевагами розробленого застосунку.

3.За результатами аналізу предметної галузі та аналізу аналогів, визначено функціональні та нефункціональні вимоги до застосунку.

4.Досліджено інструменти для розробки мобільного застосунку, внаслідок чого були обрані Flutter, Android Studio, Firebase, Ollama, llama3.2, Figma та GitHub.

5.Програмно реалізовано застосунок з використання обраних інструментів розробки та архітектурного патерну MVC.

6.Проведено тестування застосунку на відповідність вимогам. Виявлені недоліки під час тестування було виправлено.

Робота пройшла апробацію на наступних наукових конференціях:

1. Дзюба Д.Р. Савіцький В.А. Аналіз популярних фреймворків для використання в мобільній розробці // Матеріали VI Науково-технічної конференції «Сучасний стан та перспективи розвитку IoT». Збірник тез. 15 квітня 2025 року, ДУІКТ, м.Київ, (подано до друку).
2. Дзюба Д.Р. Савіцький В.А. Розробка мобільного застосунку з тестами з іноземної мови з використанням технологій штучного інтелекту // Матеріали IV Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в ІКТ». Збірник тез. 24 квітня 2025 року, ДУІКТ, м.Київ, С. 292-294.
3. Дзюба Д.Р. Савіцький В.А. Використання сервісу Firebase для захисту інформації в мобільному застосунку // Матеріали IV Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в ІКТ». Збірник тез. 24 квітня 2025 року, ДУІКТ, м.Київ, С. 436-438.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. D C. B. P. Duolingo 101: How to learn a language on Duolingo. Duolingo Blog. URL: https://blog.duolingo.com/duolingo-101-how-to-learn-a-language-on-duolingo/.

2. Getting started | lingvist. Lingvist. URL: https://lingvist.com/help/getting-started/.

3. Using babbel. Babbel. URL: https://support.babbel.com/hc/en-gb/categories/19329089895442-Using-Babbel.

4. Буровицька Ю., Гриженко Г. Мобільні технології для вивчення англійської мови майбутніми ІТ-спеціалістами // Молодь і ринок. – 2023. – № 4/212. – С. 123–128. – DOI: https://doi.org/10.24919/2308-4634.2023.279655

5. Best practice guides. Figma. URL: https://www.figma.com/best-practices/guides/.

6. Mobile application design for learning digital engineering based on figma and android studio. *Journal of computer science, information technology and telecommunication engineering*. 2023. Vol. 4. URL: https://doi.org/10.30596/jcositte.v4i1.13184

7. Топ-5 IDE і редакторів коду Python: огляд, плюси та мінуси. Sigma Software University. URL: https://university.sigma.software/best-python-ide-and-code-editors/

8. Meet android studio | android developers. Android Developers. URL: https://developer.android.com/studio/intro

9. Dart documentation. Dart programming language | Dart. URL: https://dart.dev/docs

10. Urathal, U., Vinodhini, P., & Sasikala, V. (2021). An Interpretation of Dart Programming Language. Dogo Rangsang Research Journal, 11(10), 144–149. Retrieved from https://www.researchgate.net/publication/358661479_AN_INTERPRETATION_OF_

DART_PROGRAMMING_LANGUAGE

11. Flutter documentation. Flutter. URL: https://docs.flutter.dev

12. OPUS 4 | using google´s flutter framework for the development of a large-scale reference application. OPUS 4 | ePublications TH Koeln. URL: https://epb.bibl.th-koeln.de/frontdoor/index/index/docId/1498

13. Firebase documentation. Firebase. URL: https://firebase.google.com/docs

14. Chougale P., Yadav V., Gaikwad A. Firebase – overview and usage // International Research Journal of Modernization in Engineering, Technology and Science. – 2021. – Т. 3, № 12. – С. 1178–1181. – e-ISSN 2582-5208. – URL: https://www.researchgate.net/publication/362539877_FIREBASE_-OVERVIEW_AND_USAGE

15. GitHub.com help documentation. GitHub Docs. URL: https://docs.github.com/en

16. View of PROSPECTS OF USING GIT IN THE EDUCATIONAL AND PROFESSIONAL SPACE OF UKRAINE. SWorld Journal. URL: https://www.sworldjournal.com/index.php/swj/article/view/swj27-00-047/4961

17. Gruber J. B., Weber M. rollama: An R package for using generative large language models through Ollama // arXiv.org. – 2024. – 11 квітня. – URL: https://arxiv.org/abs/2404.07654

18. Lin H., Safi T. Ollamar: An R package for running large language models. Journal of open source software. 2025. Vol. 10, no. 105. P. 7211. URL: https://doi.org/10.21105/joss.07211 (date of access: 20.05.2025).

19. Khalila Z., Nasution A. H., Monika W., Onan A., Murakami Y., Radi Y. B. I., Osmani N. M. Investigating retrieval-augmented generation in Quranic studies: a study of 13 open-source large language models // International Journal of Advanced Computer Science and Applications. – 2025. – Vol. 16, No. 2. – DOI: https://doi.org/10.14569/IJACSA.2025.01602134

20. Material design. Material Design. URL: https://m3.material.io

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка мобільного застосунку з тестами з іноземної мови з використанням технологій штучного інтелекту

Виконав студент 4 курсу
групи ПД-44
Дзюба Данило Романович
Керівник роботи
викладач кафедри ІПЗ Савіцький Вячеслав Андрійович

Київ – 2025

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи - прискорення вивчення іноземних мов в мобільному застосунку.
Об'єкт дослідження - процес вивчення іноземних мов.
Предмет дослідження - програмне забезпечення для вивчення іноземних мов.

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Проаналізувати предметну галузь застосунків для вивчення іноземних мов.
2. Провести аналіз аналогів.
3. Визначити вимоги до застосунку.
4. Дослідити та вибрати інструменти для розробки застосунку.
5. Програмно реалізувати застосунок.
6. Провести тестування застосунку.

АНАЛІЗ АНАЛОГІВ

Показник	Duolingo	Lingvist	Babbel	Розроблений проект
Платформа	Веб-сайт, IOS, Android	Веб-сайт, IOS, Android	Веб-сайт, IOS, Android	Android
Чат-бот	+	+	+	+
ШІ словник	-	-	-	+
Інтелектуальне пояснення помилок в тестах	-	-	-	+

ВИМОГИ

Функціональні вимоги:

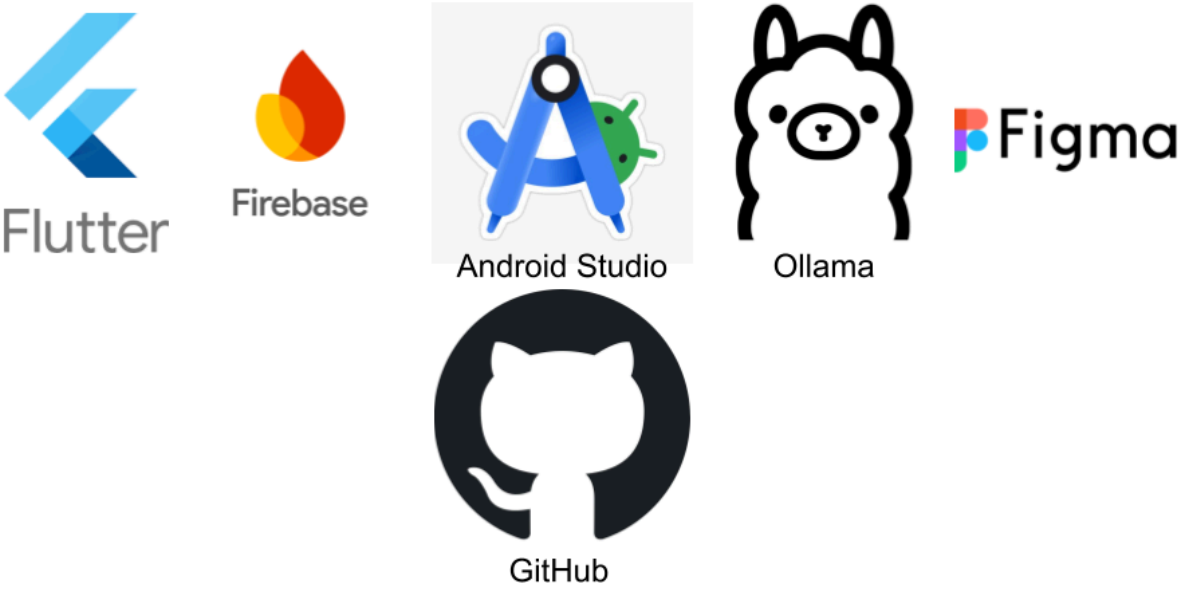
- Реєстрація та автентифікація користувачів.
- Вибір мови для тестування.
- Проходження тестів у форматі вибору правильної відповіді
- Розбивка тестів на категорії за темами.
- Використання технологій штучного інтелекту для аналізу відповідей та надання пояснень до помилок.
- Реалізувати чат-бот для практики переписки.
- Створити словник, який за допомогою штучного інтелекту буде надавати пояснення слова.

Нефункціональні вимоги:

- Сумісність із операційною системою Android.
- Час завантаження сторінок до 3 секунд.
- Забезпечення захисту персональних даних користувачів за допомогою використання Firebase Authentication.
- Інтерфейс мобільного застосунку повинен бути адаптивним до розмірів екрану.

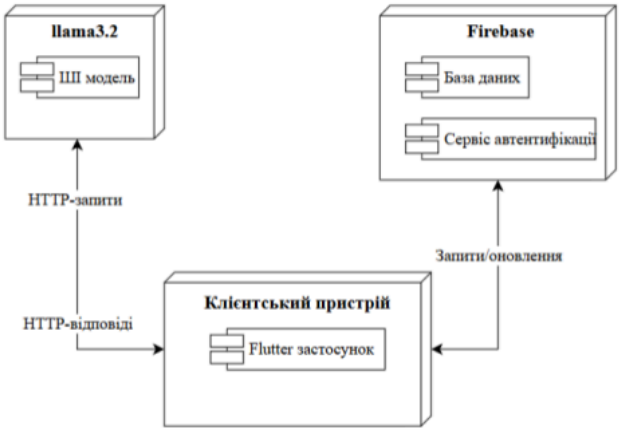
5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ



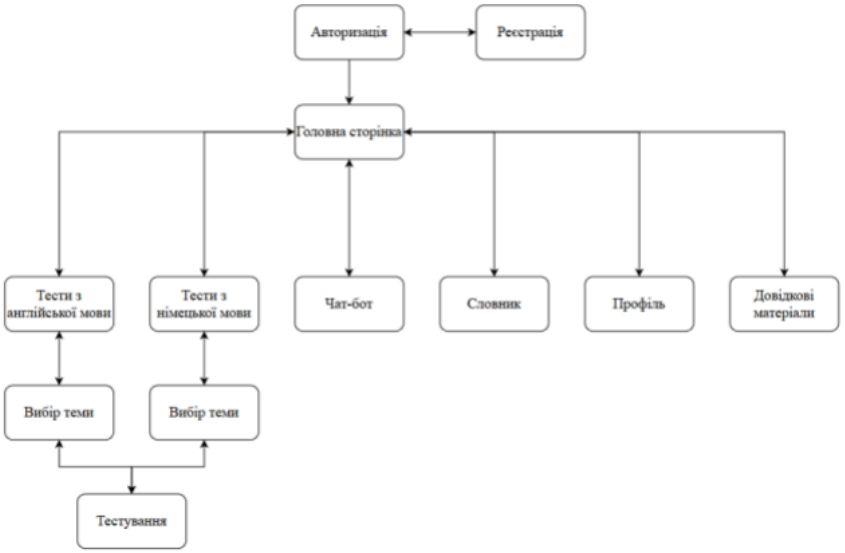
6

АРХІТЕКТУРА ЗАСТОСУНКУ



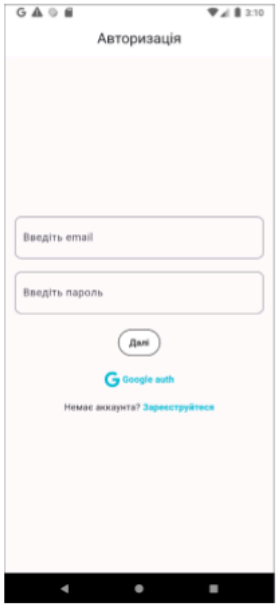
МОДЕЛЮВАННЯ ФУНКЦІОНАЛЬНОСТІ

КАРТА РОЗРОБЛЕНОГО ЗАСТОСУНКУ

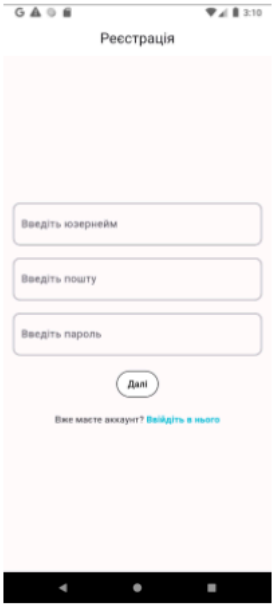


9

ЕКРАННІ ФОРМИ



Екран авторизації

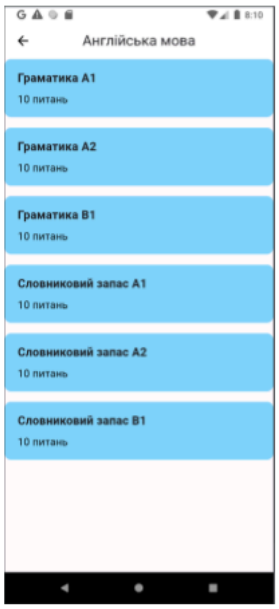


Екран реєстрації

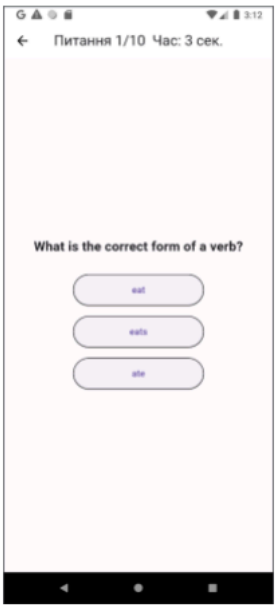
Головний екран

10

ЕКРАННІ ФОРМИ



Екран вибору теми

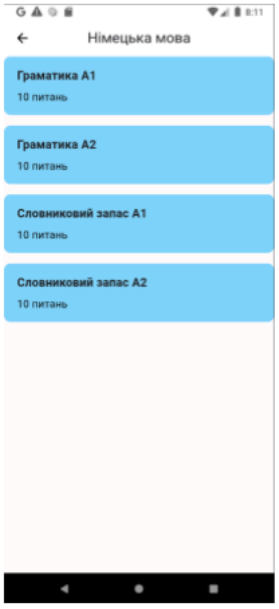


Екран тестування

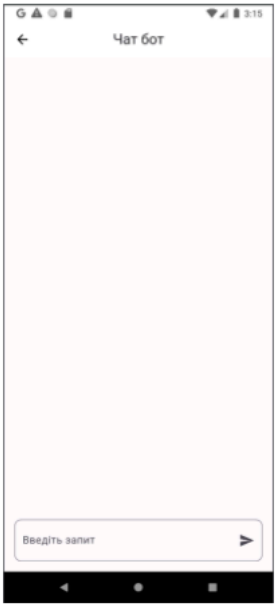
Результуючий екран

11

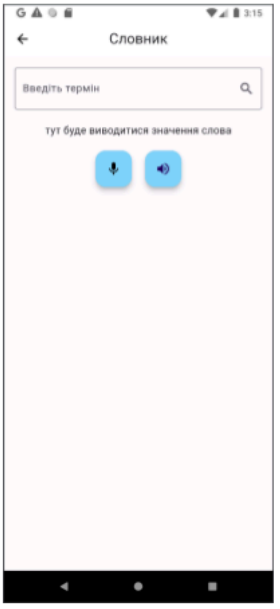
ЕКРАННІ ФОРМИ



Екран вибору теми



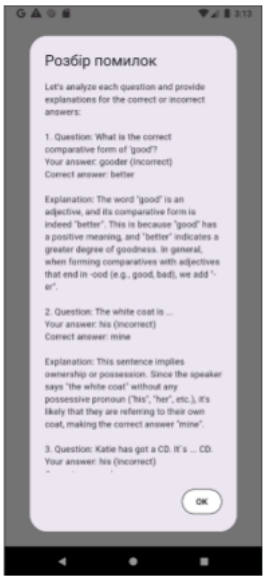
Екран чат-боту



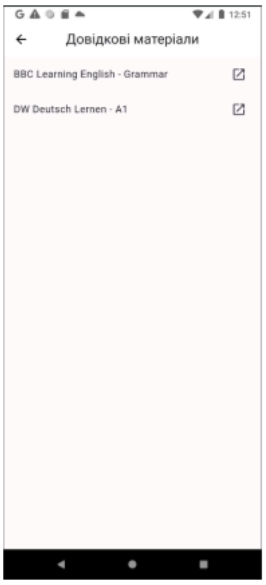
Екран словника

12

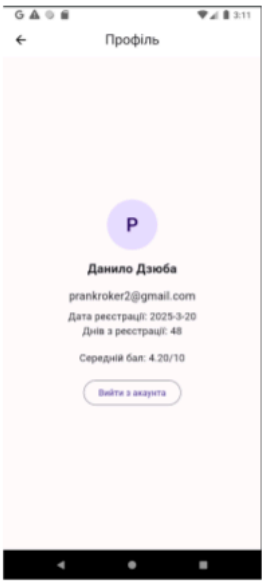
ЕКРАННІ ФОРМИ



Екран розбору помилок

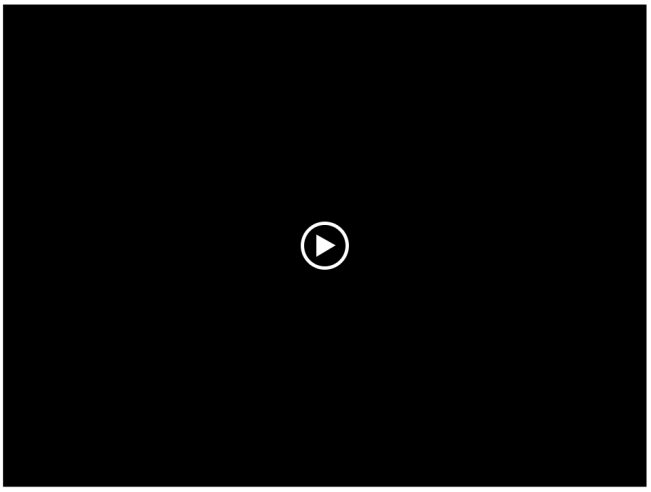


Екран з довідковими матеріалами



Екран профіля

ВІДЕО РОБОТИ ЗАСТОСУНКУ



АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Тези доповідей:

1. Дзюба Д.Р. Савіцький В.А. Аналіз популярних фреймворків для використання в мобільній розробці // VI Науково-технічна конференція «Сучасний стан та перспективи розвитку IoT». Збірник тез. 15 квітня 2025 року, ДУІКТ, м.Київ (прийнято до публікації).
2. Дзюба Д.Р. Савіцький В.А. Розробка мобільного застосунку з тестами з іноземної мови з використанням технологій штучного інтелекту // Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ». Збірник тез. 24 квітня 2025 року, ДУІКТ, м.Київ с. 292-294.
3. Дзюба Д.Р. Савіцький В.А. Використання сервісу Firebase для захисту інформації в мобільному застосунку // Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ». Збірник тез. 24 квітня 2025 року, ДУІКТ, м.Київ с. 436-438.

ВИСНОВКИ

1. Проаналізовано предметну галузь, у результаті чого підтверджено актуальність розробки мобільного застосунку та встановлено, що існує сталий попит на інструменти для самонавчання з використання штучного інтелекту.
2. Проведено аналіз аналогів, а саме Duolingo, Lingvist, Babbel. Виявлено їх обмеження, які будуть конкурентними перевагами розробленого застосунку.
3. За результатами аналізу предметної галузі та аналізу аналогів, визначено функціональні та нефункціональні вимоги до застосунку.
4. Досліджено інструменти для розробки мобільного застосунку, внаслідок чого були обрані Flutter, Android Studio, Firebase, Ollama, llama3.2, Figma та GitHub.
5. Програмно реалізовано застосунок з використання обраних інструментів розробки та архітектурного патерну MVC.
6. Проведено тестування застосунку на відповідність вимогам. Виявлені недоліки під час тестування було виправлено.

ДОДАТОК Б. ЛІСТИНГИ ОСНОВНИХ МОДУЛІВ

Вихідний код файлу main.dart

```
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import
'package:flutter_diploma/global/animation/splashScreen.dart'
;
import 'package:flutter_diploma/themes/theme.dart';
import 'package:flutter_dotenv/flutter_dotenv.dart';

Future main() async {//підключення бази даних і запуск
програми
 WidgetsFlutterBinding.ensureInitialized();
 await dotenv.load(fileName: 'assets/.env');
 if(kIsWeb) {
  await Firebase.initializeApp(
     options: FirebaseOptions(apiKey:
dotenv.env["API_KEY"]??"",
       appId: dotenv.env["APP_ID"]??"",
       messagingSenderId: dotenv.env["MSG_ID"]??"",
       projectId: "mobilka-ai-feat")
  );
 }
 else{
  await Firebase.initializeApp();
 }
 runApp(const MyHome());
}

class MyHome extends StatelessWidget {
 const MyHome({super.key});

 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   debugShowCheckedModeBanner: false,//відключення
прапорця дебаг
    theme: buildAppTheme(),// застосуваня тем
    home: const SplashScreen(),// виклик класу та його
запуск
   );
 }
}
```

Вихідний код файлу TopicListScreen.dart

```
import 'package:flutter/material.dart';
import
'package:flutter_diploma/controllers/TopicController.dart';
import 'package:flutter_diploma/themes/theme.dart';

class TopicListScreen extends StatelessWidget {
 final String language;
 final TopicController controller;

 TopicListScreen({
  super.key,
  required this.language,
 }) : controller = TopicController(language: language);

 @override
 Widget build(BuildContext context) {
  return Theme(
   data: buildAppTheme(),
   child: Scaffold(
    appBar: AppBar(
     title: Text(language),
     centerTitle: true,
    ),
    body: _buildTopicList(),
   ),
  );
 }

 Widget _buildTopicList() {
  return ListView.builder(
   itemCount: controller.topics.length,
   itemBuilder: (context, index) {
    final topic = controller.topics[index];
    return Padding(
     padding: const EdgeInsets.only(bottom: 16.0),
      child: GestureDetector(
       onTap: () => controller.handleTopicTap(context,
topic.id),
        child: Container(
         decoration: BoxDecoration(
          color: Colors.lightBlue[200],
          borderRadius: BorderRadius.circular(10),
         ),
         padding: const EdgeInsets.symmetric(vertical: 16,
horizontal: 20),
         child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
           Text(
            topic.title,
            style: const TextStyle(
              fontSize: 18,
              fontWeight: FontWeight.bold),
           ),
           const SizedBox(height: 8),
           Text(
            "${topic.questions} питань",
            style: const TextStyle(fontSize: 16, color:
Colors.black),
           )
          ],
         ),
        ),
       );
      },
     );
    }
}
```

Вихідний код файлу germanyQuestionsPage.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_diploma/views/TopicListScreen.dart';
```

```
class GermanQuestions extends StatelessWidget {
  const GermanQuestions({super.key});

  @override
  Widget build(BuildContext context) {
    return TopicListScreen(language: "Німецька мова");
  }
}
```

Вихідний код файлу Theory.dart

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

class Theory extends StatelessWidget{

  final List<Map<String,String>> materials = [
    {
      'title':'BBC Learning English - Grammar',

'url':'https://www.bbc.co.uk/learningenglish/english/grammar'
    },
    {
      'title':'DW Deutsch Lernen - A1',

'url':'https://learngerman.dw.com/en/beginners/s-62078399'
    }
  ];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar:
        AppBar(title: const Text('Довідкові матеріали'),
          centerTitle: true,),
        body: ListView.builder(
          itemCount: materials.length,
            itemBuilder: (context,index){
            final item = materials[index];
            return ListTile(
              title: Text(item['title']!),
              trailing: const Icon(Icons.open_in_new),
              onTap: () async {
                final url = Uri.parse(item['url']!);
                if (await canLaunchUrl(url)) {
                await launchUrl(url);
                }
              },
            );
            })
      );
    }
}
```

Вихідний код файлу searchWord.dart

```
import 'package:flutter/material.dart';
import
'package:flutter_diploma/controllers/SearchController.dart';

class SearchScreen extends StatefulWidget {
  final SearchWordController controller;
  const SearchScreen({super.key, required this.controller});

  @override
  State<SearchScreen> createState() =>
_SearchScreenState();
}

class _SearchScreenState extends State<SearchScreen> {
  @override
  void initState() {
    super.initState();
    widget.controller.init();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Словник"),
        centerTitle: true,
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: ListView(
          children: [
            TextField(
              controller: widget.controller.textController,
              decoration: InputDecoration(
                labelText: "Введіть термін",
                border: const OutlineInputBorder(),
                suffixIcon: IconButton(
                  icon: const Icon(Icons.search),
                  onPressed: widget.controller.handleSearch,
                ),
              ),
            ),
            const SizedBox(height: 20),
            ValueListenableBuilder<bool>(
              valueListenable:
widget.controller.isLoadingNotifier,
              builder: (_, isLoading, __) {
                return isLoading
                    ? const Center(child:
CircularProgressIndicator())
                    : ValueListenableBuilder<String>(
                      valueListenable:
widget.controller.wordMeaningNotifier,
                      builder: (_, meaning, __) {
                        return Text(
                          meaning,
                          style: const TextStyle(fontSize: 16),
                          textAlign: TextAlign.center,
                        );
                      },
                    );
              },
            ),
            const SizedBox(height: 20),
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                FloatingActionButton(
                  heroTag: "Mic",
                  backgroundColor: Colors.lightBlue[200],
                  onPressed: widget.controller.handleMicPress,
                  child: const Icon(Icons.mic, color: Colors.black),
```

```
            ),
            const SizedBox(width: 20),
            ValueListenableBuilder<bool>(
              valueListenable:
widget.controller.isSpeakingNotifier,
              builder: (_, isSpeaking, __) {
                return FloatingActionButton(
                  heroTag: "Speak",
                  backgroundColor: Colors.lightBlue[200],
                  onPressed: widget.controller.handleSpeak,
                  child: Icon(
                    isSpeaking ? Icons.volume_off :
Icons.volume_up,
```

Вихідний код файлу quizPage.dart

```
import 'package:flutter/material.dart';
import
'package:flutter_diploma/controllers/QuizController.dart';
import
'package:flutter_diploma/views/screens/mainPage.dart';

class QuizPage extends StatefulWidget {
  final String topicId;
  const QuizPage({super.key, required this.topicId});

  @override
  State<QuizPage> createState() => _QuizPageState();
}

class _QuizPageState extends State<QuizPage> {
  late final QuizController controller;

  @override
  void initState() {
    super.initState();
    controller = QuizController(widget.topicId);
    controller.onTimeTick = () => setState(() {});
    controller.loadQuestions().then((_) {
      setState(() {});
      controller.startTimer();
    });
  }

  @override
  void dispose() {
    controller.stopTimer();
    super.dispose();
  }

  void _handleAnswer(int selectedIndex) async {
    final isFinished =
controller.answerQuestion(selectedIndex);
    setState(() {});
    if (isFinished) {
      controller.stopTimer();
      await controller.saveResult();
      _showResults();
    }
  }

  void _showResults() async {
    await showDialog(
      context: context,
      barrierDismissible: false,
      builder: (_) => AlertDialog(
        title: const Text("Результати"),
        content: Text(
            "Ваш результат: ${controller.score} з
${controller.questions.length}\nЧас:
${controller.elapsedTime} сек."),
        actions: [
          if (controller.mistakes.isNotEmpty)
            TextButton(
              onPressed: () {
                Navigator.of(context).pop();
                _showMistakeAnalysis();
              },
              child: const Text("Розбір помилок"),
            ),
          TextButton(
            onPressed: () {
              Navigator.of(context).pop();
              Navigator.pushReplacement(
                context,
                MaterialPageRoute(builder: (_) => const
MainPage()),
              );
            },
            child: const Text("OK"),
          ),
        ],
      ),
    );
  }

  void _showMistakeAnalysis() async {
    showDialog(
      context: context,
      barrierDismissible: false,
      builder: (_) => const AlertDialog(
        title: Text("Аналіз помилок"),
        content: Center(child: CircularProgressIndicator()),
      ),
    );

    final result = await controller.analyzeMistakes();

    if (!mounted) return;
    Navigator.of(context).pop(); // закрити завантаження
    showDialog(
      context: context,
      builder: (_) => AlertDialog(
        title: const Text("Розбір помилок"),
        content: SingleChildScrollView(child: Text(result)),
        actions: [
          TextButton(
            onPressed: () {
              Navigator.of(context).pop();
              Navigator.pushReplacement(
                context,
                MaterialPageRoute(builder: (_) => const
MainPage()),
              );
            },
            child: const Text("OK"),
```

```
        ),
       ],
     ),
    );
  }

  @override
  Widget build(BuildContext context) {
    if (controller.questions.isEmpty) {
      return const Scaffold(
        body: Center(child: CircularProgressIndicator()),
      );
    }

    if (controller.currentIndex >= controller.questions.length) {
      return const Scaffold(
        body: Center(child: Text("Тест завершено.")),
      );
    }

    final question =
controller.questions[controller.currentIndex];

    return Scaffold(
      appBar: AppBar(
        title: Text(
          "Питання ${controller.currentIndex +
1}/${controller.questions.length}  Час:
${controller.elapsedTime} сек.",
        ),
        centerTitle: true,
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text(
                question.question,
                textAlign: TextAlign.center,
                style: const TextStyle(fontSize: 20, fontWeight:
FontWeight.bold),
              ),
              const SizedBox(height: 20),
              ...List.generate(question.options.length, (index) {
                return Padding(
                  padding: const EdgeInsets.symmetric(vertical:
8.0),
                  child: ElevatedButton(
                    onPressed: () => _handleAnswer(index),
                    child: Text(question.options[index]),
                  ),
                );
              }),
            ],
          ),
        ),
      ),
    );
  }
}
```

Вихідний код файлу profilePage.dart

```
import 'package:flutter/material.dart';
import
'package:flutter_diploma/controllers/ProfileController.dart';
import 'package:flutter_diploma/models/UserModel.dart';
import
'package:flutter_diploma/views/screens/auth/loginPage.dart';

class Profile extends StatefulWidget {
  const Profile({super.key});

  @override
  State<Profile> createState() => _ProfileState();
}

class _ProfileState extends State<Profile> {
  final ProfileController _controller = ProfileController();
  late Future<ProfileModel> _profileFuture;
  bool _isLoading = false;

  @override
  void initState() {
    super.initState();
    _profileFuture = _controller.loadProfile();
  }

  String _formatDate(DateTime? date) {
    return date != null
        ? "${date.year}-${date.month}-${date.day}"
        : "Невідомо";
  }

  int _calculateDays(DateTime? date) {
    return date != null
        ? DateTime.now().difference(date).inDays
        : 0;
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Профіль'),
        centerTitle: true,
        elevation: 0,
      ),
      body: FutureBuilder<ProfileModel>(
        future: _profileFuture,
        builder: (context, snapshot) {
          if (snapshot.connectionState ==
ConnectionState.waiting) {
            return const Center(child:
CircularProgressIndicator());
          }

          if (snapshot.hasError || !snapshot.hasData) {
            return const Center(child: Text("Помилка
завантаження даних"));
          }

          final profile = snapshot.data!;
          final days = _calculateDays(profile.registrationDate);

          return Center(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                CircleAvatar(
                  radius: 40,
                  child: Text(
                    profile.email?.substring(0, 1).toUpperCase() ??
"?",
                    style: const TextStyle(fontSize: 30),
```

```
            ),
          ),
          const SizedBox(height: 16),
          Text(
            profile.username ?? "Без імені",
            style: const TextStyle(
              fontSize: 20,
              fontWeight: FontWeight.bold
            ),
          ),
          const SizedBox(height: 16),
          Text(profile.email ?? "", style: const
TextStyle(fontSize: 18)),
          const SizedBox(height: 8),
          Text(
            'Дата реєстрації:
${_formatDate(profile.registrationDate)}',
            style: const TextStyle(fontSize: 16),
          ),
          Text(
            'Днів з реєстрації: $days',
            style: const TextStyle(fontSize: 16),
          ),
          const SizedBox(height: 20),
          Text(
            "Середній бал:
${profile.averageScore.toStringAsFixed(2)}/10",
            style: const TextStyle(fontSize: 16),
          ),
          const SizedBox(height: 20),
          OutlinedButton(
            onPressed: () async {
              setState(() => _isLoading = true);
              await _controller.logout();
              Navigator.pushReplacement(context,
MaterialPageRoute(builder: (context) => const Login()));
            },
            child: _isLoading
                ? const CircularProgressIndicator()
                : const Text('Вийти з акаунта'),
          ),
        ],
      ),
    );
  },
 ),
);
 }
}
```

Вихідний код файлу mainPage.dart

```
import 'package:flutter/material.dart';
import
'package:flutter_diploma/controllers/ChatController.dart';
import
'package:flutter_diploma/controllers/SearchController.dart';
import 'package:flutter_diploma/models/ChatModel.dart';
import 'package:flutter_diploma/models/SearchModel.dart';
import 'package:flutter_diploma/themes/theme.dart';
import 'package:flutter_diploma/views/screens/Theory.dart';
import
'package:flutter_diploma/views/screens/chatBotPage.dart';
import
'package:flutter_diploma/views/screens/profilePage.dart';
import
'package:flutter_diploma/views/screens/searchWord.dart';
import
'package:flutter_diploma/views/widgets/englishQuestionPage
.dart';
import
'package:flutter_diploma/views/widgets/germanyQuestionsPa
ge.dart';
import
'package:font_awesome_flutter/font_awesome_flutter.dart';

class MainPage extends StatefulWidget {
 const MainPage({super.key});

 @override
 State<MainPage> createState() => _MainPageState();
}

class _MainPageState extends State<MainPage> {

 final ChatController _chatController =
ChatController(ChatModel());
 final SearchWordController _searchWordController =
SearchWordController(SearchModel());
 @override
 Widget build(BuildContext context) {
  const String appTitle = 'Тести';

  return Theme(
    data: buildAppTheme(), // Використання теми з файлу
    child: Scaffold(
     appBar: AppBar(
      automaticallyImplyLeading: false,
      title: const Text(appTitle),
      centerTitle: true,
     ),
     body: Center(
      child: Padding(
       padding: const EdgeInsets.all(16),
       child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
         ElevatedButton.icon(
          onPressed: () {
           Navigator.push(context,
MaterialPageRoute(builder: (context) => const
EnglishQuestions()));
          },
          icon: const Icon(
           FontAwesomeIcons.flagUsa,
           color: Colors.blue,
          ),
          label: const Text('Англійська мова'),
         ),
         const SizedBox(height: 20),
         ElevatedButton.icon(
          onPressed: () {
           Navigator.push(context,
MaterialPageRoute(builder: (context) => const
GermanQuestions()));
          },
          icon: const Icon(
           Icons.abc,
           color: Colors.purple,
          ),
          label: const Text('Німецька мова'),
         ),
         const SizedBox(height: 20),
         ElevatedButton.icon(onPressed: () {
          Navigator.push(context,
MaterialPageRoute(builder: (context) =>
ChatScreen(controller: _chatController)));
          },
```

```
            icon: const Icon(
              Icons.chat,
              color: Colors.purple
            ),
            label:const Text('Чат-бот'),
          ),
        ],
      ),
    ),
  ),
  bottomNavigationBar: BottomNavigationBar(
    items: const [
      BottomNavigationBarItem(icon: Icon(Icons.search),
label: 'Словник'),
      BottomNavigationBarItem(icon: Icon(Icons.book),
label: 'Теорія'),
      BottomNavigationBarItem(icon: Icon(Icons.person),
label: 'Профіль'),
    ],
    onTap: (index) {
      // Обробка натискання на елементи навігації
      if(index == 0){
        Navigator.push(context, MaterialPageRoute(builder:
(context) => SearchScreen(controller:
_searchWordController,)));
      }
      if(index == 1){
        Navigator.push(context, MaterialPageRoute(builder:
(context) => Theory()));
      }
      if(index == 2){
        Navigator.push(context, MaterialPageRoute(builder:
(context) => const Profile()));
      }
    },
  ),
 );
 }
}
```

Вихідний код файлу chatBotPage.dart

```
import 'package:flutter/material.dart';
import
'package:flutter_diploma/controllers/ChatController.dart';

class ChatScreen extends StatefulWidget {
 final ChatController controller;
 const ChatScreen({super.key, required this.controller});

 @override
 State<ChatScreen> createState() => _ChatScreenState();
}

class _ChatScreenState extends State<ChatScreen> {
 final TextEditingController _textController =
TextEditingController();

 @override
 void initState() {
  super.initState();
  // Слухаємо зміни в моделі
  widget.controller.model.messagesStream.listen((_) {
   setState(() {});
  });
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    title: const Text("Чат бот"),
    centerTitle: true,
   ),
   body: Column(
    children: [
     Expanded(
      child: ListView.builder(
       itemCount:
widget.controller.model.messages.length,
       itemBuilder: (context, index) {
        if (index == 0) return const SizedBox.shrink();
        final message =
widget.controller.model.messages[index];
        return _buildMessageBubble(message);
       },
      ),
     ),
     Padding(
      padding: const EdgeInsets.all(16.0),
      child: TextField(
       controller: _textController,
       onSubmitted: (text) => _handleSend(),
       decoration: InputDecoration(
        labelText: "Введіть запит",
        suffixIcon: IconButton(
         icon: const Icon(Icons.send),
         onPressed: _handleSend,
        ),
       ),
      ),
     ),
    ],
   ),
  );
 }

 Widget _buildMessageBubble(Map<String, String>
message) {
  return Container(
   margin: const EdgeInsets.all(8),
   padding: const EdgeInsets.all(12),
   decoration: BoxDecoration(
    color: message["role"] == 'assistant' ? Colors.blue[100] :
Colors.green[100],
    borderRadius: BorderRadius.circular(12),
   ),
   child: Text(message["content"] ?? ''),
  );
 }

 void _handleSend() {
  if (_textController.text.isNotEmpty) {
   widget.controller.handleUserInput(_textController.text);
   _textController.clear();
  }
 }
}
```

Вихідний код файлу loginPage.dart

```
import 'package:flutter/material.dart';
import
'package:flutter_diploma/controllers/LoginController.dart';
import 'package:flutter_diploma/themes/theme.dart';
import
'package:flutter_diploma/views/screens/auth/registrationPage
.dart';
import
'package:font_awesome_flutter/font_awesome_flutter.dart';

class Login extends StatefulWidget{
  const Login({super.key});

  @override
  State<Login> createState() => _LoginState();
}

class _LoginState extends State<Login> {

  final LoginController loginController = LoginController();
  final emailController = TextEditingController();
  final passwordController = TextEditingController();

  @override
  void dispose() {
    emailController.dispose();
    passwordController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    const String appTitle = 'Авторизація';

    return Theme(
      data: buildAppTheme(),
      child: Scaffold(
        appBar: AppBar(
          automaticallyImplyLeading: false,
          title: const Text(appTitle),
          centerTitle: true,
        ),
        body: Center(
          child: Padding(
            padding: const EdgeInsets.all(16),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                TextField(
                  key: const ValueKey('email_field'),
                  controller: emailController,
                  decoration: const InputDecoration(
                    labelText: 'Введіть email',
                  ),
                ),
                const SizedBox(height: 20,),
                TextField(
                  key: const ValueKey('password_field'),
                  controller: passwordController,
                  decoration: const InputDecoration(
                    labelText: 'Введіть пароль',
                  ),
                ),
                const SizedBox(height: 20,),
                TextButton(
                  onPressed:(){

loginController.signIn(context,emailController.text,password
Controller.text);
                  },
                  child: const Text("Далі"),
                ),
                const SizedBox(height: 20,),
                Row(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: [
                    const Icon(FontAwesomeIcons.google,color:
Colors.cyan,),
                    const SizedBox(height: 5,),
                    GestureDetector(
                      onTap: (){
                        loginController.signInWithGoogle(context);
                      },
                      child: const Text(" Google auth",style:
TextStyle(color: Colors.cyan,fontWeight: FontWeight.bold),),
                    )
                  ],
                ),
                const SizedBox(height: 20,),
                Row(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: [
                    const Text("Немає аккаунта?"),
                    const SizedBox(height: 5,),
                    GestureDetector(
                      onTap: (){
                        Navigator.push(context,
MaterialPageRoute(builder: (context) => const
Registration()));
                      },
                      child: const Text(" Зареєструйтеся",style:
TextStyle(color: Colors.cyan,fontWeight: FontWeight.bold),),
                    )
                  ],
                ),
              ],
            ),
          ),
        ),
      );
    }
}
```

Вихідний код файлу user_repository.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';

class UserRepository {
  Future<void> saveUserToFirestore(User user) async {
    DocumentReference userDoc =
FirebaseFirestore.instance.collection("users").doc(user.uid);

    DocumentSnapshot docSnapshot = await userDoc.get();
    if (!docSnapshot.exists) {
      await userDoc.set({
        'username': user.displayName ?? "Google User",
        'email': user.email,
      });
    }
  }
}
```

Вихідний код файлу firebase_auth_services.dart

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter_diploma/global/common/toast.dart';

class FirebaseAuthService{

  final FirebaseAuth _auth = FirebaseAuth.instance;

  Future<User?>
signUpWithEmailAndPassword(BuildContext context, String
email, String password) async {

    if(email.isEmpty || password.isEmpty){
      showToast(context: context, message: 'пошта або пароль
не мають бути пусті');
      return null;
    }

    try{
      UserCredential credential = await
_auth.createUserWithEmailAndPassword(email: email,
password: password);
      return credential.user;
    }on FirebaseAuthException catch(e){
      if(e.code == 'email-already-in-use'){
        showToast(context: context, message: 'дана пошта вже
використовується');
      }else if(e.code == 'invalid-email'){
        showToast(context: context, message: 'пошта в
неправильному форматі');
      }else if(e.code == 'weak-password'){
        showToast(context: context, message: 'слабкий
пароль');
      }else{
        showToast(context: context, message: 'Error:
${e.code}');
      _wordMeaning = "Введіть термін для пошуку.";
      notifyListeners();
      return;
    }
      }
    }
    return null;
  }

  Future<User?>
signInWithEmailAndPassword(BuildContext context, String
email, String password) async {

    if(email.isEmpty || password.isEmpty){
      showToast(context: context, message: 'пошта або пароль
не мають бути пусті');
      return null;
    }

    try{
      UserCredential credential = await
_auth.signInWithEmailAndPassword(email: email, password:
password);
      return credential.user;
    }on FirebaseAuthException catch(e){
      if(e.code == 'user-not-found' || e.code ==
'wrong-password' || e.code == 'invalid-credential' || e.code ==
'invalid-email'){
        showToast(context: context, message: 'Неправильний
пароль або пошта');
      }else{
        showToast(context: context, message: 'Error:
${e.code}');
      }
    }
    return null;
  }
}
```

Вихідний код файлу QuizModel.dart

```
class QuizQuestion {
  final String question;
  final List<String> options;
  final int correctIndex;

  QuizQuestion({
    required this.question,
    required this.options,
    required this.correctIndex,
  });

  factory QuizQuestion.fromMap(Map<String, dynamic>
map) {
    return QuizQuestion(
      question: map['question'],
      options: List<String>.from(map['options']),
      correctIndex: map['correctIndex'],
    );
  }
}

class Mistake {
  final String question;
  final String selectedAnswer;
  final String correctAnswer;

  Mistake({
    required this.question,
    required this.selectedAnswer,
    required this.correctAnswer,
  });

  Map<String, String> toMessageFormat() => {
    "question": question,
    "selectedAnswer": selectedAnswer,
    "correctAnswer": correctAnswer,
  };
}
```

Вихідний код файлу ChatModel.dart

```
import 'dart:async';
import 'dart:convert';
import 'package:http/http.dart' as http;

class ChatModel {
  final List<Map<String, String>> messages = [
    {"role": "system", "content": "You are a helpful assistant
for user to improve skills in language. You need to point out
all their mistakes and explain how to get better, while
handling conversation"},
```

```
  ];

  // Стрім для сповіщень про зміни
  final StreamController<List<Map<String, String>>>
_messagesController =
  StreamController.broadcast();

  Stream<List<Map<String, String>>> get messagesStream
=> _messagesController.stream;

  Future<void> sendMessage(String prompt) async {
    final userMessage = {"role": "user", "content": prompt};
    messages.add(userMessage);
    _messagesController.add([...messages]); // Сповістити
про зміни

    try {
      final response = await http.post(
        Uri.parse("http://10.0.2.2:11434/api/chat"),
        headers: {"Content-Type": "application/json"},
        body: json.encode({
          "model": "llama3.2",
          "messages": messages,
          "stream": false,
        }),
      );

      if (response.statusCode == 200) {
        final data = json.decode(response.body);
        messages.add({"role": "assistant", "content":
data["message"]["content"]});
        _messagesController.add([...messages]); // Сповістити
про оновлення
      } else {
        messages.removeLast();
        _messagesController.add([...messages]);
      }
    } catch (e) {
      messages.removeLast();
      _messagesController.add([...messages]);
    }
  }

  void dispose() {
    _messagesController.close();
  }
}
```

Вихідний код файлу toast.dart

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

void showToast({required BuildContext context,required
String message}){
  ScaffoldMessenger.of(context).hideCurrentSnackBar(); //
Прибирає попередній
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text(message),
      backgroundColor: Colors.cyan,
      behavior: SnackBarBehavior.floating,
      duration: const Duration(seconds: 2),
    ),
  );
}
```

Вихідний код файлу TopicController.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_diploma/models/TopicModel.dart';
import
'package:flutter_diploma/views/screens/quizPage.dart';

class TopicController {
  final String language;
  final List<Topic> topics;

  TopicController({required this.language}) : topics =
_loadTopics(language);

  static List<Topic> _loadTopics(String language) {
    switch (language) {
      case 'Німецька мова':
        return [
          Topic(id: "GrammarGermany", title: "Граматика A1",
questions: 10),
          Topic(id: "GrammarGermanyA2", title: "Граматика
A2", questions: 10),
          Topic(id: "VocabularGermany", title: "Словниковий
запас A1", questions: 10),
          Topic(id: "VocabularGermanyA2", title:
"Словниковий запас A2", questions: 10)
        ];
      case 'Англійська мова':
        return [
          Topic(id: "Grammar", title: "Граматика A1",
questions: 10),
          Topic(id: "GrammarA2", title: "Граматика A2",
questions: 10),
          Topic(id: "GrammarB1", title: "Граматика B1",
questions: 10),
          Topic(id: "Vocabulary", title: "Словниковий запас
A1", questions: 10),
          Topic(id: "VocabularyA2", title: "Словниковий запас
A2", questions: 10),
          Topic(id: "VocabularyB1", title: "Словниковий запас
B1", questions: 10)
        ];
      default:
        return [];
    }
  }

  void handleTopicTap(BuildContext context, String topicId)
{
    Navigator.push(
      context,
      MaterialPageRoute(
      builder: (context) => QuizPage(topicId: topicId),
    ));
  }
}
```

Вихідний код файлу RegistrationController.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
```

```
import 'package:flutter/material.dart';
import 'package:flutter_diploma/global/common/toast.dart';
import
'package:flutter_diploma/services/firebase_auth_services.dart
';
import
'package:flutter_diploma/views/screens/auth/loginPage.dart';

class RegistrationController {

  final FirebaseAuthService _auth = FirebaseAuthService();

  void signUp(BuildContext context,String username,String
email,String password) async{

    User? user = await
_auth.signUpWithEmailAndPassword(context, email,
password);
    if(user != null){
      await
FirebaseFirestore.instance.collection("users").doc(user.uid).s
et({
        'username':username,
        'email':email
       });

      showToast(context:context, message: "User is succesfully
created");
      Navigator.push(context, MaterialPageRoute(builder:
(context) => const Login()));
    }
  }
}
```

Вихідний код файлу QuizController.dart

```
import 'dart:async';
import 'dart:convert';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter_diploma/models/QuizModel.dart';
import 'package:http/http.dart' as http;

class QuizController {
  final String topicId;
  final List<QuizQuestion> questions = [];
  final List<Mistake> mistakes = [];
  int currentIndex = 0;
  int score = 0;
  int elapsedTime = 0;
  Timer? _timer;
  final user = FirebaseAuth.instance.currentUser;
  int? _selectedAnswer;

  int? get selectedAnswer => _selectedAnswer;
  set selectedAnswer(int? value) => _selectedAnswer = value;
  Function()? onTimeTick;

  QuizController(this.topicId);

  Future<void> loadQuestions() async {
    final snapshot = await FirebaseFirestore.instance
        .collection('topics')
        .doc(topicId)
        .collection('questions')
        .get();

    questions.addAll(snapshot.docs.map((e) =>
QuizQuestion.fromMap(e.data())));
  }

  void startTimer() {
    _timer = Timer.periodic(const Duration(seconds: 1), (_) {
      elapsedTime++;
      onTimeTick?.call();
    });
  }

  void stopTimer() {
    _timer?.cancel();
  }

  bool answerQuestion(int selectedIndex) {
    if (currentIndex >= questions.length) return true; //
запобігає виходу за межі масиву

    final question = questions[currentIndex];
    if (selectedIndex == question.correctIndex) {
      score++;
    } else {
      mistakes.add(Mistake(
        question: question.question,
        selectedAnswer: question.options[selectedIndex],
        correctAnswer: question.options[question.correctIndex],
      ));
    }
    currentIndex++;
    return currentIndex >= questions.length;
  }

  Future<void> saveResult() async {
    await FirebaseFirestore.instance.collection('results').add({
      'uid': user?.uid,
      'topic': topicId,
      'score': score,
      'totalQuestions': questions.length,
      'timeSpent': elapsedTime,
      'timestamp': FieldValue.serverTimestamp(),
    });
  }

  Future<String> analyzeMistakes() async {
    final List<Map<String, String>> messages = [
      {
        "role": "system",
        "content":
        "Explain and analyze strictly in English only, no matter
what language is used in the questions or answers (e.g.,
German, Spanish, etc.). For each answer, explain why it is
correct or incorrect, and provide the correct answer when
necessary. Do not use any language other than English in
your explanation."
      },
      {
        "role": "user",
        "content": _formatMistakes(),
      }
    ];

    final Map<String, Object> data = {
      "model": "llama3.2",
      "messages": messages,
      "stream": false,
      "max_tokens": 100,
    };

    try {
      final res = await http.post(
```

```
      Uri.parse("http://10.0.2.2:11434/api/chat"),
      headers: {"Content-Type": "application/json"},
      body: json.encode(data),
    );

    if (res.statusCode == 200) {
      final decoded = json.decode(res.body);
      return decoded["message"]["content"] ?? "Помилка
отримання відповіді.";
    } else {
      return "Не вдалося отримати пояснення від AI.";
    }
  } catch (e) {
    return "Сталася помилка: $e";
  }
 }

 String _formatMistakes() {
  return mistakes.map((m) {
    return "Question: ${m.question}\nYour answer:
${m.selectedAnswer}\nCorrect answer:
${m.correctAnswer}\n";
  }).join("\n");
 }
}
```

Вихідний код файлу ProfileController.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter_diploma/models/UserModel.dart';

class ProfileController {
 final FirebaseAuth _auth = FirebaseAuth.instance;
 final FirebaseFirestore _firestore =
FirebaseFirestore.instance;

 Future<ProfileModel> loadProfile() async {
  final user = _auth.currentUser;
  if (user == null) throw Exception("User not logged in");

  // Отримання даних користувача
  final userDoc = await
_firestore.collection('users').doc(user.uid).get();
  final resultsSnapshot = await _firestore.collection('results')
      .where('uid', isEqualTo: user.uid)
      .get();

  // Обчислення середнього балу
  double averageScore = 0.0;
  if (resultsSnapshot.docs.isNotEmpty) {
   final total = resultsSnapshot.docs.fold(0, (sum, doc) {
    return sum + (doc.data()['score'] as int? ?? 0);
   });
   averageScore = total / resultsSnapshot.docs.length;
  }

  return ProfileModel(
   uid: user.uid,
   email: user.email,
   username: userDoc.data()?['username'],
   registrationDate: user.metadata.creationTime,
   averageScore: averageScore,
  );
 }

 Future<void> logout() async {
  await _auth.signOut();
 }
}
```