МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри_____ Є. О. Давиденко

підпис

«___» _____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Мобільний застосунок вивчення сурдоалфавіту з використанням технології розпізнавання жестів

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 408.22011014

Здобувачка _____ Н. О. Морозова

підпис

«__» _____ 2024 р.

Керівник PhD, ст. викладач кафедри ІПЗ _____ К. О. Антіпова

підпис

«__» _____ 2024 р.

Консультант канд. техн. наук, доцент _____ А. О. Алексєєва

підпис

«__» _____ 2024 р.

Миколаїв - 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ
Зав. кафедри Є. О. Давиденко
« » 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи бакалавра

Видано здобувачці групи 408 факультету комп'ютерних наук

Морозовій Надії Олексіївні

(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи

Мобільний застосунок вивчення сурдоалфавіту з використанням технології розпізнавання жестів

Затверджена наказом по ЧНУ від «22» грудня 2023 р. № 269

2. Строк представлення кваліфікаційної роботи «___» червня 2024 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Очікуваним результатом є мобільний застосунок вивчення сурдоалфавіту з впровадженням технології розпізнавання жестів

4. Перелік питань, що підлягають розробці:

- дослідження предметної області та аналіз існуючих аналогів;
- формування специфікації вимог до мобільного застосунку;
- моделювання і проєктування користувацького інтерфейсу для програмного забезпечення;
- розробка архітектури програмного забезпечення;
- тестування мобільного застосунку, зокрема перевірка коректності розпізнавання зображень.

5. Перелік графічних матеріалів

Презентація

6. Завдання до спеціальної частини

Аналіз охорони праці в лабораторії ЧНУ

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алєксєєва А. О.	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи PhD, ст. викладач кафедри ІПЗ Антіпова Катерина Олександрівна

(посада, прізвище, ім'я, по батькові)

(підпис)

Завдання прийнято до виконання

Морозова Надія Олексіївна

(прізвище, ім'я, по батькові студента)

(підпис)

Дата видачі завдання «_» _____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема:__ Мобільний застосунок вивчення сурдоалфавіту з використанням технології розпізнавання жестів_____
_____

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	15.12.2023	22.12.2023	Виконано
2.	Огляд літератури згідно теми КРБ	20.01.2024	30.01.2024	Виконано
3.	Складання календарного плану КРБ	01.02.2024	02.02.2024	Виконано
4.	Аналіз предметної області	05.02.2024	08.02.2024	Виконано
5.	Розробка проєктних рішень до КРБ	12.02.2024	15.02.2024	Виконано
6.	Проєктування інтерфейсу користувача	16.02.2024	18.03.2024	Виконано
7.	Розробка функціональної частини застосунку	18.03.2024	08.04.2024	Виконано
8.	Інтеграція технології розпізнавання жестів	09.04.2024	22.04.2024	Виконано
9.	Проведення тестування розробленого застосунку	23.04.2024	26.04.2024	Виконано
10.	Розробка окремого модулю з охорони праці	27.04.2024	10.05.2024	Виконано
11.	Оформлення КРБ та презентації для захисту	13.05.2024	30.05.2024	Виконано
12.	Відгук керівника КРБ	31.05.2024	31.05.2024	Виконано
13.	Предзахист КРБ	03.06.2024	03.06.2024	Виконано

14.	Виправлення зауважень в результаті попереднього захисту КРБ	04.06.2024	10.06.2024	Виконано
15.	Подання рецезенту КРБ	10.06.2024	10.06.2024	Виконано
16.	Відгук рецезента КРБ	11.06.2024	11.06.2024	Виконано
17.	Захист КРБ	27.06.2024	28.06.2024	

Розробила здобувачка Морозова Надія Олексіївна _____

(*прізвище, ім'я, по батькові)* *(підпис)*

«__»_____2024 р.

Керівник роботи Ph.D., ст. викладач кафедри ІПЗ Антіпова Катерина Олександрівна

_____

(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«__»_____2024 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Мобільний застосунок вивчення сурдоалфавіту з використанням технології розпізнавання жестів»

Здобувачка 408 гр.: Морозова Надія Олексіївна

Керівник: Ph.D., ст. викладач Антіпова К.О.

Актуальність теми даної кваліфікаційної роботи визначається в тому що, в сучасному світі все більше людей цікавляться вивченням мови жестів, особливо через зростання усвідомлення її важливості для спілкування з глухими або нездатними до мовлення особами. Зростання інтересу до цієї теми визначається не лише потребою взаємодії з особами з обмеженими можливостями, а й прагненням вдосконалити комунікацію та розуміння між людьми різних культур та мов. Технології розпізнавання жестів, які базуються на штучному інтелекті, робить процес навчання легшим та додає інтерактиву для того, щоб користувач краще міг засвоїти матеріал. Також, розробка мобільного застосунку з використанням технології розпізнавання жестів забезпечує доступне навчання для всіх бажаючих, незалежно від їхніх фізичних можливостей. Крім того, вона може покращити користувацький досвід та зробити процес вивчення мови жестів більш захоплюючим та ефективним.

Кваліфікаційна робота присвячена розробці мобільного застосунку вивчення сурдоалфавіту з використанням технології розпізнавання жестів, який забезпечує доступне навчання для всіх бажаючих, незалежно від їхніх фізичних можливостей

Об’єктом роботи є процес вивчення сурдоалфавіту з використанням технології розпізнавання жестів.

Предметом кваліфікаційної роботи є інструменти розробки мобільних застосунків та технології розпізнавання жестів із використанням штучного інтелекту.

Метою кваліфікаційної роботи є розробка мобільного застосунку вивчення алфавіту мови жестів для забезпечення вдосконалення комунікації та розуміння між людьми різних здібностей.

Для досягнення поставленої мети необхідно виконати наступний перелік завдань:

1) дослідити предметну галузь згідно отриманої теми КРБ;

2) розробити інтерфейс мобільного застосунку, який зрозумілим та зручним для користувачів будь-якого віку та можливостей;

3) проєктування та моделювання архітектури мобільного застосунку;

4) інтегрувати технологію розпізнавання жестів для можливості більш детального вивчення сурдоалфавіту;

5) забезпечити безпеку та конфіденційність даних користувачів, які використовують застосунок;

6) провести тестування та вдосконалення застосунку згідно з отриманими результатами.

Структура кваліфікаційної роботи бакалавра включає вступ, чотири розділи, висновки та перелік джерел посилань.

Вступ окреслює важливість теми, формулює мету, дає короткий огляд задачі, а також описує предмет, об'єкт та методи дослідження.

У першому розділі було проаналізовано наявні методи, що використовуються у галузі розробки мобільних застосунків для вивчення сурдоалфавіту. На підставі цього аналізу були сформульовані функціональні вимоги до мобільного застосунку, що визначають його основні можливості та вимоги до взаємодії з користувачем.

У другому розділі окреслюється процес створення функціональних та інформаційних моделей мобільного застосунка з використанням технології розпізнавання зображень. Спочатку визначаються основні функції застосунку, які необхідні для забезпечення зручного та ефективного користування. Наступний крок: розробка інформаційної моделі, яка відображає структуру та взаємозв'язки між різними компонентами застосунку.

У третьому визначається архітектура мобільного застосунку, вибираються необхідні технології та мови програмування для реалізації функціоналу. Визначаються ключові компоненти застосунку, їх взаємодія та структура, розробляються діаграми, які відображають архітектурні рішення та логіку роботи застосунку. У висновках виконується оцінка виконаної роботи та здобутих результатів.

У четвертому розділі відбувається реалізація програмного забезпечення відповідно до розробленої архітектури та моделей. Спочатку створюються окремі модулі та компоненти застосунку, які потім інтегруються в єдину систему. Після цього проводиться тестування застосунку з метою перевірки його працездатності та відповідності вимогам.

Кваліфікаційна робота бакалавра викладена на 76 сторінок, вона містить 4 розділи, 42 ілюстрації, 9 таблиць, 23 джерел в переліку посилань.

Ключові слова: *мобільний застосунок, сурдоалфавіт, розробка, програмування на Kotlin, розробка під Android, розпізнавання жестів, штучний інтелект.*

ABSTRACT

of the Bachelor’s Thesis

"Mobile application for learning sign language alphabet using gesture recognition technology"

Student of group 408: Morozova Nadiia Oleksiivna

Supervisor: Ph.D., Senior Lecturer Antipova K. O.

The relevance of the topic of this qualification work is determined by the increasing interest in learning sign language, especially due to the growing awareness of its importance for communication with deaf or speech-impaired individuals. The growing interest in this topic is determined not only by the need to interact with individuals with disabilities but also by the desire to improve communication and understanding between people of different cultures and languages. Gesture recognition technologies based on artificial intelligence make the learning process easier and add interactivity to help users better absorb the material. Additionally, developing a mobile application using gesture recognition technology provides accessible learning for all interested parties, regardless of their physical abilities. Moreover, it can enhance the user experience and make the process of learning sign language more engaging and effective.

This work is dedicated to the development of a mobile application for learning the sign language alphabet using gesture recognition technology, which provides accessible learning for all interested parties, regardless of their physical abilities.

The object of the work is the process of learning the sign language alphabet using gesture recognition technology.

The subject of the qualification work is mobile application development tools and gesture recognition technology using artificial intelligence.

The purpose of the qualification work is to develop a mobile application for learning sign language alphabet to improve communication and understanding between people with different abilities.

To achieve the stated goal, the following tasks need to be accomplished:

1) research the subject area according to the received topic of the qualification work;

2) develop a user-friendly interface for the mobile application suitable for users of all ages and abilities;

3) design and model the architecture of the mobile application;

4) integrate gesture recognition technology to allow for a more detailed study of the sign language alphabet;

5) ensure the security and confidentiality of user data using the application;

6) conduct testing and improvement of the application based on the obtained results.

The structure of the bachelor's qualification work includes an introduction, four chapters, conclusions, and a list of references.

The introduction outlines the importance of the topic, formulates the goal, provides a brief overview of the task, and describes the subject, object, and research methods.

The first chapter analyzes existing methods used in the field of mobile application development for learning the sign language alphabet. Based on this analysis, functional requirements for the mobile application are formulated, determining its main features and user interaction requirements.

The second chapter outlines the process of creating functional and informational models of the mobile application using image recognition technology. Initially, the main functions necessary for convenient and effective use are determined. The next step is the development of an informational model that reflects the structure and relationships between different components of the application.

The third chapter defines the architecture of the mobile application, selects the necessary technologies and programming languages for implementation. Key components of the application, their interaction, and structure are determined, and diagrams reflecting architectural decisions and the logic of the application's operation are developed. The conclusions evaluate the work done and the results obtained.

The fourth chapter involves the implementation of software according to the developed architecture and models. Initially, individual modules and components of the application are created, which are then integrated into a single system. Subsequently, the application is tested to verify its performance and compliance with requirements.

The bachelor's qualification work consists of 76 pages, including 4 chapters, 42 illustrations, 9 tables, and 23 sources in the reference list.

Keywords: *mobile application, sign language alphabet, development, Kotlin programming, Android development, gesture recognition, artificial intelligence.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних

КРБ – кваліфікаційна робота бакалавра

ОС – операційна система

ПЗ – програмне забезпечення

ШІ – штучний інтелект

API – application programming interface

CRUD – create, read, update, delete

HTTP – Hypertext Transfer Protocol

ML – machine learning

MVVM – model-view-view model

SQL – structured query language

UI – user interface

ВСТУП

Актуальність теми кваліфікаційної роботи бакалавра полягає в тому, що в сучасному світі все більше людей цікавляться вивченням мови жестів, особливо через зростання усвідомлення її важливості для спілкування з глухими або нездатними до мовлення особами. Зростання інтересу до цієї теми визначається не лише потребою взаємодії з особами з обмеженими можливостями, а й прагненням вдосконалити комунікацію та розуміння між людьми різних культур та мов. Технології розпізнавання жестів, які базуються на штучному інтелекті, робить процес навчання легшим та додає інтерактиву для того, щоб користувач краще міг засвоїти матеріал. Також, розробка мобільного застосунку з використанням технології розпізнавання жестів забезпечує доступне навчання для всіх бажаючих, незалежно від їхніх фізичних можливостей. Крім того, вона може покращити користувацький досвід та зробити процес вивчення мови жестів більш захоплюючим та ефективним.

Об’єктом роботи є процес вивчення сурдоалфавіту з використанням технології розпізнавання жестів.

Предметом кваліфікаційної роботи є інструменти розробки мобільних застосунків та технології розпізнавання жестів із використанням штучного інтелекту.

Метою кваліфікаційної роботи є розробка мобільного застосунку вивчення алфавіту мови жестів для забезпечення вдосконалення комунікації та розуміння між людьми різних здібностей.

Для досягнення поставленої мети необхідно виконати наступний перелік завдань:

1) дослідити предметну галузь згідно отриманої теми КРБ;

2) розробити інтерфейс мобільного застосунку, який зрозумілим та зручним для користувачів будь-якого віку та можливостей;

3) проєктування та моделювання архітектури мобільного застосунку;

4) інтегрувати технологію розпізнавання жестів для можливості більш детального вивчення сурдоалфавіту;

5) забезпечити безпеку та конфіденційність даних користувачів, які використовують застосунок;

6) провести тестування та вдосконалення застосунку згідно з отриманими результатами.

Кваліфікаційна робота бакалавра викладена на 76 сторінок, містить 4 розділи, 9 таблиць, 42 ілюстрації, 23 джерела в переліку посилань.

Ключові слова: *мобільний застосунок, сурдоалфавіт, розробка, програмування на Kotlin, розробка під Android, розпізнавання жестів, штучний інтелект.*

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд аналогів мобільного застосунку вивчення сурдоалфавіту

При розробці мобільного застосунку вивчення сурдоалфавіту вкрай є важливим дослідження аналогів. Адже саме завдяки цьому можна зрозуміти, який саме функціонал є найбільш популярним або які недоліки мають конкуренти для того щоб уникнути тих самих помилок. Для аналізу обрані такі 3 мобільні застосунки: ASL Bloom — Sign Language (табл. 1.1), Sign Language ASL Pocket Sign (табл. 1.2), Lingvano: Sign Language – ASL (табл. 1.3). Розглянемо кожний аналог окремо.

Таблиця 1.1 – Опис аналогу «ASL Bloom — Sign Language»

Назва	ASL Bloom — Sign Language
Виробник	SignLab
Архітектура	Клієнт-сервер
Мова реалізації	Kotlin
Основні функції, характеристики	1) великий словник, вікторини та діалоги; 2) поради щодо граматики та культури; 3) вивчення мови за певними модулями; 4) є контент за підпискою; 5) сторінка «My Account»; 6) відстежування прогресу по окремим урокам або за певний проміжок часу; 7) алфавіт/мова жестів вивчається завдяки анімованими відео викладачів.
Переваги	1) мобільний застосунок доступний у будь-який час та будь-якому місці; 2) велика кількість інтерактивних уроків та вправ.

Недоліки	Відсутність функції розпізнавання жестів у реальному часі за допомогою ШІ
Джерело інформації (вебсайт)	https://play.google.com/store/apps/details?id=com.toleio.us

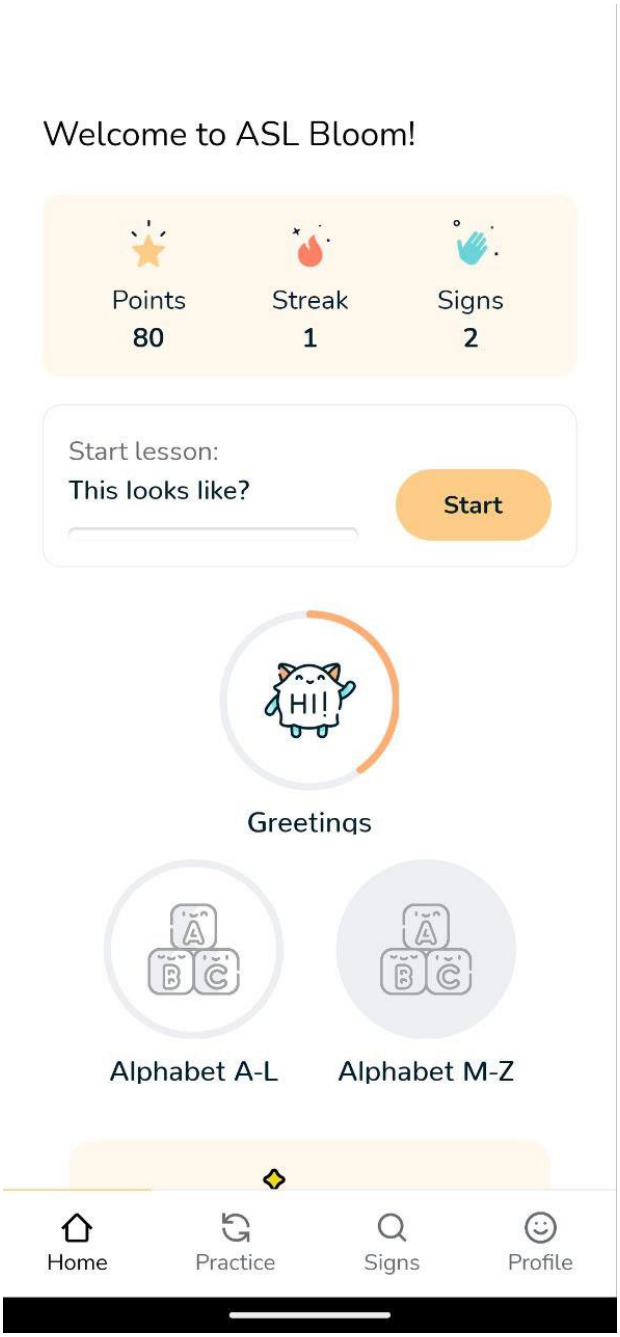

Рисунок 1.1 – Головна сторінка «ASL Bloom — Sign Language»

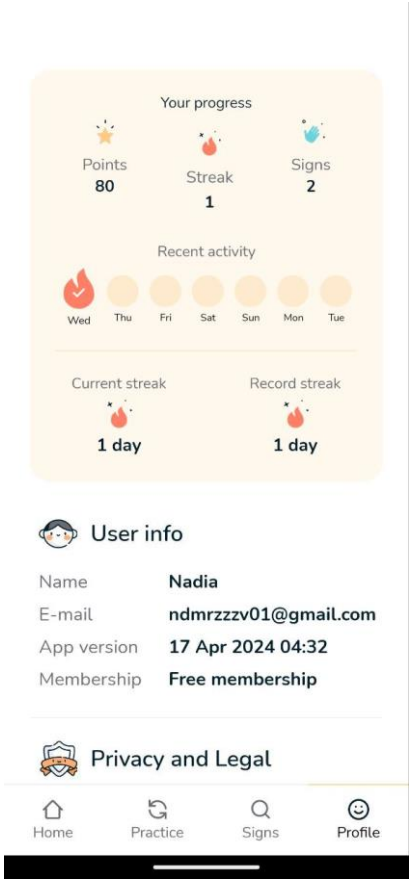

Рисунок 1.2 – Сторінка персонального акаунту користувача у «ASL Bloom — Sign Language»

Таблиця 1.2 – Опис аналогу «Sign Language ASL Pocket Sign»

Назва	Sign Language ASL Pocket Sign
Виробник	MobiReactor
Архітектура	Клієнт-сервер
Мова реалізації	Kotlin
Основні функції, характеристики	1) великий словник, вікторини та діалоги; 2) поради щодо граматики та культури; 3) вивчення мови за певними модулями; 4) є контент за підпискою; 5) відстежування прогресу по окремим урокам;

	6) алфавіт/мова жестів вивчається завдяки анімованими відео викладачів; 7) керування підписками.
Переваги	1) мобільний застосунок доступний у будь-який час та будь-якому місці; 2) велика кількість інтерактивних уроків та вправ.
Недоліки	1) відсутність функції розпізнавання жестів у реальному часі за допомогою ШІ; 2) дизайн, котрий ніяк не привертаю увагу користувача та не дає бажання навчатися алфавіту.
Джерело інформації (вебсайт)	https://play.google.com/store/apps/details?id=com.mobirectactor.signlanguage

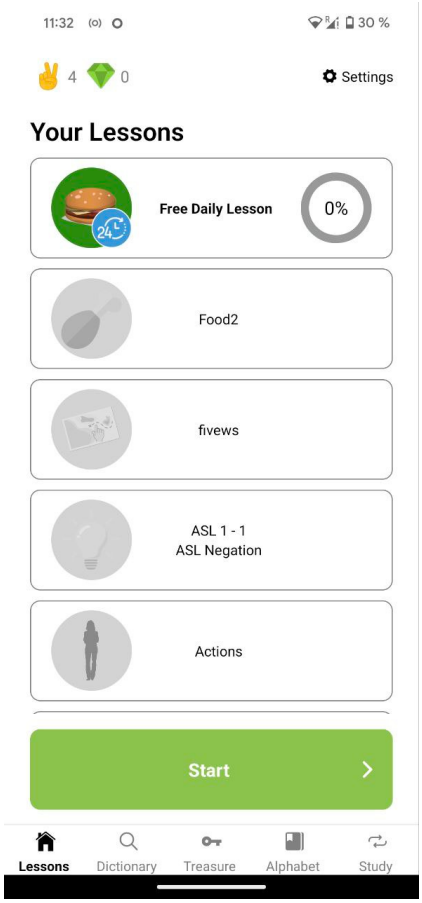

Рисунок 1.3 – Головна сторінка «Sign Language ASL Pocket Sign»

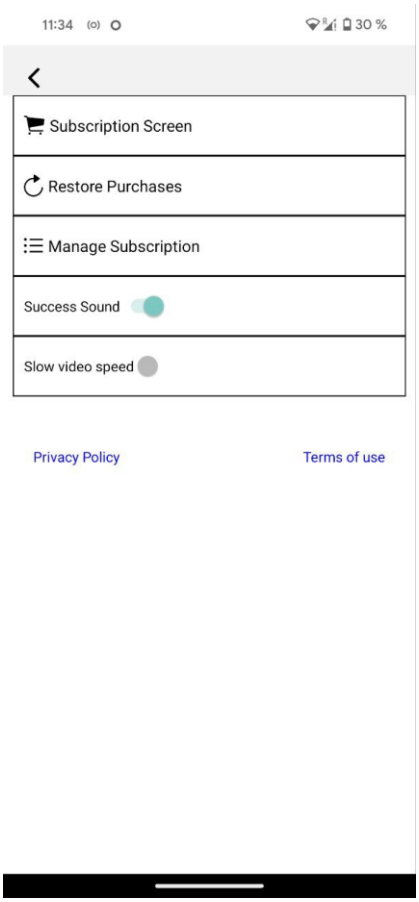

Рисунок 1.4 – Сторінка налаштувань «Sign Language ASL Pocket Sign»

Таблиця 1.3 – Опис аналогу «Lingvano: Sign Language – ASL»

Назва	Lingvano: Sign Language – ASL
Виробник	Lingvano
Архітектура	Клієнт-сервер
Мова реалізації	Kotlin
Основні функції, характеристики	1) великий словник, вікторини та діалоги; 2) поради щодо граматики та культури; 3) є контент за підпискою; 4) сторінка «My Account»; 5) відстежування прогресу по окремим урокам;

	6) алфавіт/мова жестів вивчається завдяки анімованими відео викладачів.
Переваги	1) мобільний застосунок доступний у будь-який час та будь-якому місці; 2) велика кількість інтерактивних уроків та вправ.
Недоліки	1) відсутність функції розпізнавання жестів у реальному часі за допомогою ШІ; 2) забагато докучливої реклами.
Джерело інформації (вебсайт)	https://play.google.com/store/apps/details?id=com.lingvano.app

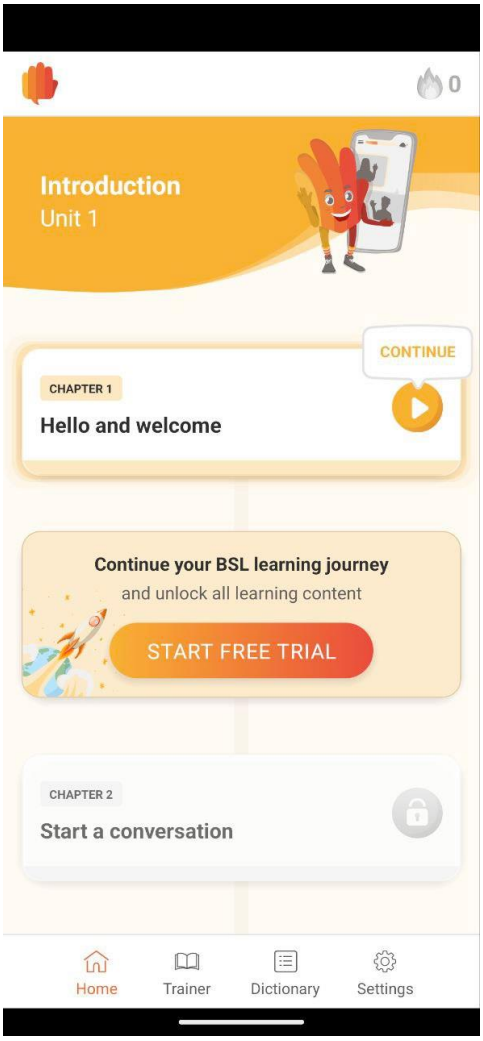

Рисунок 1.5 – Головна сторінка «Lingvano: Sign Language – ASL»

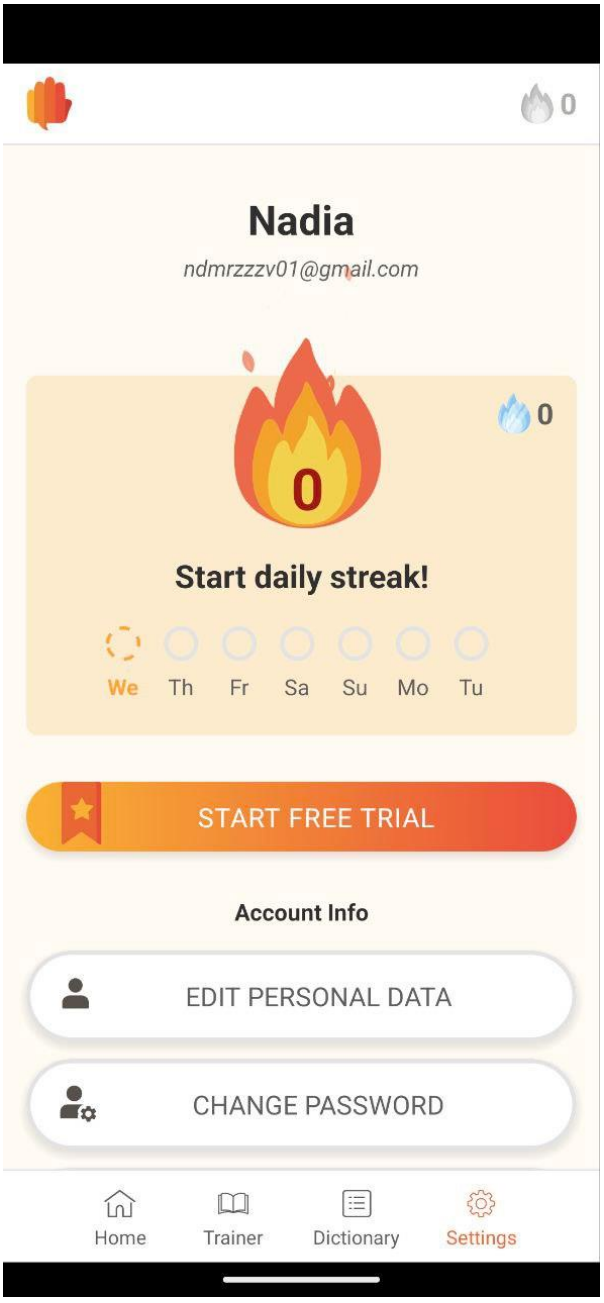

Рисунок 1.6 – Сторінка персонального акаунта користувача «Lingvano: Sign Language – ASL»

Підбиваючи підсумки, можна виокремити один головний недолік у трьох застосунках-аналогах – це те, що всі вони не мають функції розпізнавання жестів за допомогою штучного інтелекту.

1.2 Опис системи, що розробляється

Мобільний застосунок вивчення сурдоалфавіту з використанням технології розпізнавання жестів «EasyGestures» призначений для полегшення процесу навчання основ сурдоалфавіту шляхом інтерактивної взаємодії з користувачем. Завдяки використанню передових технологій розпізнавання жестів, програма дозволяє користувачам вивчати абетку жестів в зручному мобільному форматі. Цей застосунок може бути корисним як для людей, які вивчають сурдоалфавіт вперше, так і для тих, хто хоче покращити свої навички. Він надає можливість візуального навчання, а також інтерактивного тренування, що сприяє кращому засвоєнню матеріалу.

Крім того, застосунок «EasyGestures» має різноманітні функції, які полегшують навчання та сприяють його ефективності. Наприклад, програма може включати інтерактивні вправи та ігри, що допомагають користувачам підвищити швидкість та точність виконання жестів. Також, застосунок може вести статистику прогресу користувача, дозволяючи відстежувати його досягнення та вдосконалювати навички на основі індивідуальних потреб. Тож, розглянемо детальний опис системи, що розробляється.

Основні функції «EasyGestures»:

1) реєстрація нових користувачів;
2) авторизація користувачів;
3) перегляд усіх доступних курсів вивчення мови жестів;
4) перегляд детальної інформації про певний курс;
5) можливість додавати курс в список улюблених;
6) пошук курсів за назвою;
7) пошук курсів за категорією;
8) автоматичне розпізнавання жесту за допомогою технології;
9) перегляд особистої інформації;
10) перегляд особистого прогресу у вивченні курсів;

11) вивчення уроку у вигляді вікторини;

12) можливість повторного проходження курсу/уроку;

13) можливість скасування поточного прогресу.

Ролі користувачів системи:

Гість: Може переглядати список уроків та курсів, реєструватися або авторизуватися.

Зареєстрований користувач: Має всі можливості гостя, а також може вивчати алфавіт мови жестів, проходити курси та отримати детальну персональну інформації.

Розглянемо декілька **основних сценаріїв роботи системи**:

Сценарій 1 – Реєстрація

Мета – Створити новий профіль у застосунку

Сценарій:

1) користувач відкриває застосунок та бачить Splash Screen (це графічний елемент керування, що складається з вікна, що містить зображення, логотип і поточну версію ПЗ);

2) на вибір дається SignUp через пошту або через Google;

3) користувач обирає SignUp через пошту: вводить пошту та пароль і натискає кнопку «Next» та має створений новий профіль;

4) користувач обирає SignUp через Google: відкривається нове вікно, де користувач обирає профіль Google та після має створенний новий профіль.

Сценарій 2 – Вивчення матеріалу

Мета – Вивчити певну частину матеріалу через один урок в курсі

1) авторизований користувач відкриває програму, відкриває пошук та вводить назву бажаного курсу або обирає серед списку;

2) застосунок пропонує навчальні уроки, починаючи з основних символів сурдоалфавіту;

3) користувач проходить уроки, навчаючись жестам;

4) після завершення уроків користувач може пройти тести на розуміння та запам'ятовування вивченого матеріалу.

Сценарій 3 – Практична частина

Мета – Закріпити навички через тест

1) після вивчення одного уроку у користувача є можливість пройти тест для закріплення результату;

2) застосунок пропонує серію ігрових завдань, у яких користувачеві потрібно розпізнавати жести сурдоалфавіту та давати правильну відповідь серед поданих відповідей у тесті;

3) користувач може відстежувати свій прогрес та покращувати навички з кожним проходженням;

4) після невдалої спроби у користувача буде вибір: пройти урок знову або пройти тест знову.

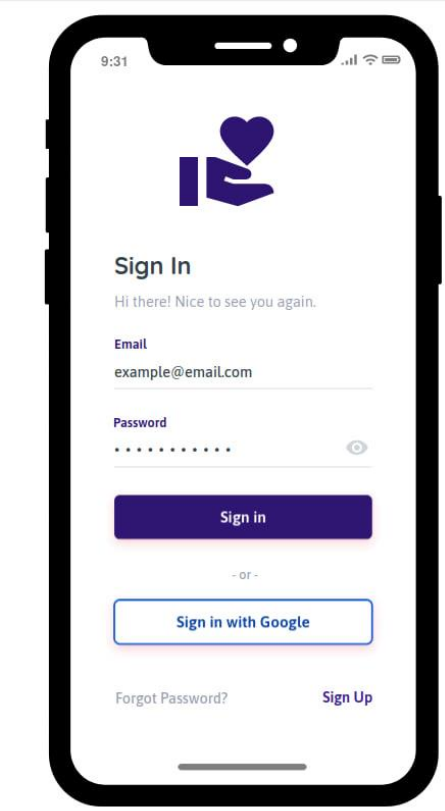

Рисунок 1.7 – Mock-up екрану авторизації (користувач не має акаунт)

Рисунок 1.8 – Mock-up головного екрану (користувач має акаунт)

Перші, приблизні mock-up створені за допомогою сервісу: https://moqups.com/. На цих двох рисунках зображено як саме приблизно буде функціонувати застосунок, а саме, що користувач буде бачити, коли не має акаунту та якщо має.

1.3 Специфікація вимог до програмного забезпечення

ПРИЗНАЧЕННЯ ТА МЕЖІ ПРОЄКТУ

Призначення застосунку, для якої розробляється програмне забезпечення

Призначення мобільного застосунку вивчення сурдоалфавіту з використанням технології розпізнавання жестів «EasyGestures» для полегшення процесу навчання основ сурдоалфавіту шляхом інтерактивної взаємодії з користувачем.

Межі проєкту ПЗ

Крайня дата завершення роботи над ПЗ – 21.05.2024

ЗАГАЛЬНИЙ ОПИС

Сфера застосування

Мобільний застосунок може бути застосований в спеціалізованих школах для людей з вадами слуху або в повсякденному житті.

Характеристика користувачів

Користувач повинен мати мобільний пристрій на базі ОС Android із доступом в Інтернет.

Загальна структура та склад системи

Android-застосунок відповідає за такі функції як: Login/Signup, Settings (Shared Preferences), My Profile, обробку query до network (через API), збереження великої кількості інформації до local storage (DB) та реалізація розпізнавання жестів за допомогою ШІ.

Загальні обмеження

Основне обмеження – доступ до Інтернету.

ФУНКЦІЇ СИСТЕМИ ANDROID-ЗАСТОСУНКУ

Функція вивчення сурдоалфавіту за допомогою розпізнавання жестів

Опис функції

Користувач може сфотографувати жест задля вивчення або покращення вивчення сурдоалфавіту.

Вхідна і вихідна інформація

Вхідна інформація: зображення жесту, яке надав користувач.

Вихідна інформація: виведення результату, чи правильно користувач показав жест.

Функція зміни фото або ім’я користувача на сторінці «My Account»

Опис функції

Користувач може змінити персональну інформацію.

Вхідна і вихідна інформація

Вхідна інформація: персональна інформація, така як фото або ім’я.

Вихідна інформація: виведення нового фото або ім’я.

Функція вивчення сурдоалфавіту за письмового тесту.

Опис функції

Користувач вводить відповіді на запитання.

Вхідна і вихідна інформація

Вхідна інформація: відповіді користувача на запитання.

Вихідна інформація: виведення результату тесту.

Функціональні вимоги

Система машинного навчання для розпізнавання жестів, база даних товарів магазину та доступ до Інтернету.

ВИМОГИ ДО ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

Джерела та зміст вхідної інформації (даних)

Вхідна інформація надходить від користувачів, які використовують функції застосунку.

Вимоги до способів організації, збереження та ведення інформації

Обмін даними здійснюється через RESTful API, локальною базою даних буде Room (SQLite).

ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

Для застосунку не передбачено жорстких обмежень. Він повинен працювати на будь-якому сучасному смартфоні з підтримкою ОС Android 8.0 та вище, та об’ємом ОЗП 1 ГБ та більше.

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Архітектура програмної системи

– База даних: Реляційна база даних Room (SQLite).

– Мобільний клієнт: Застосунок для Android, розроблений на Kotlin.

Системне програмне забезпечення

– Мобільна платформа: Android 8.0 (Oreo) або вище.

– Фреймворк: Android SDK для розробки нативних застосунків.

– Бібліотеки: Retrofit для взаємодії з API, Glide/Picasso для обробки зображень жестів, Room для роботи з локальною базою даних.

Мережне програмне забезпечення

– Протоколи: Використання HTTPS для захищеного обміну даними між сервером і клієнтом.

– Обмін даними: RESTful API для взаємодії між Android-застосунком і сервером.

Програмне забезпечення для обробки зображень

– Використання спеціалізованої бібліотеки для розпізнавання жестів TensorFlow Lite.

ВИМОГИ ДО ЗОВНІШНІХ ІНТЕРФЕЙСІВ

Інтерфейс користувача

Інтерфейс повинен бути максимально зрозумілим до користувача, тому для цього було використано стандартні рішення при створені інтерфейсу мобільного застосунку.

Апаратний інтерфейс

Мобільний застосунок має бути доступним на будь-якому пристрої із будь-яким розширенням, тому створено адаптивний інтерфейс.

Програмний інтерфейс

Jetpack Compose – це сучасний набір інструментів для створення рідного інтерфейсу Android. Jetpack Compose спрощує та прискорює розробку інтерфейсу користувача на Android за допомогою меншого коду, потужних інструментів та інтуїтивно зрозумілих API Kotlin.

ВЛАСТИВОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Доступність

Доступний для будь-якого користувача, який має бажання ознайомитись із сурдоалфавітом та має мобільний пристрій на базі ОС Android.

Супроводжуваність

Мобільний застосунок не потребує супроводжуваності, окрім випадків, коли клієнт бажає змінити дизайн або додати якийсь новий функціонал.

Переносимість

Мобільний застосунок доступний на будь-якому пристрої на базі ОС Android.

Продуктивність

Продуктивність не залежить від Інтернет-з’єднання.

Надійність

Пароль користувача хешований за допомогою засобів Firebase.

Висновки до розділу 1

В першому розділі кваліфікаційної роботи бакалавра було розглянуто існуючі аналоги Android-застосунку вивчення сурдоалфавіту та, власне, проаналізовано їх. Обрано 3 застосунки-конкуренти для вивчення їх переваг та недоліків, сформовано таблиці та детальні описи по кожному з них.

Також проведено аналіз розроблюваної системи, в якому визначено різновиди користувачів, основні можливості та сценарії використання мобільного застосунку. У цьому розділі сформульовані вимоги до програмного забезпечення «EasyGestures», детально описано його призначення та функціональність, визначено архітектуру та необхідні технології для розробки застосунку. Сформовано специфікацію вимог до програмного забезпечення, в які входить: призначення та межі проєкту, загальний опис ПЗ, функції системи мобільного застосунку, вимоги до інформаційного забезпечення, вимоги до технічного забезпечення, вимоги до програмного забезпечення, вимоги до зовнішніх інтерфейсів та властивості програмного забезпечення.

2 МОДЕЛЮВАННЯ ЗАСТОСУНКУ

2.1 Сценарії використання

Use Case є описом способів, в яких користувачеві interacts з системою або продуктом. Використовуючи це може визначити значення результатів сценаріїв, помилки сценаріїв, і будь-які критичні зміни або виключення. У разі використання може бути написано або виконано візуально з допомогою методу використання моделі інструмента [1]. Use Case корисні [2]:

1) коли потрібна докладна специфікація вимог для розробки або підтримки системи, яка включає модель даних, опис інтерфейсу, інтеграцію з іншими системами та сценарії використання;

2) для виявлення та виправлення помилок у системі, дозволяючи розібратися, на якому етапі виникла проблема;

3) якщо потрібно описати функціональність чи користувацький інтерфейс системи у формі сценаріїв, юзкейси можуть бути використані як основа для цього опису. Наприклад, для мобільного застосунка основні вимоги можуть бути описані через інтерфейс, а деякі складніші функції можуть бути уточнені за допомогою таблиць або поєднанням зі сценаріями.

Розглянемо на прикладі, один із самих важливих Use Case – це реєстрація або логін користувача в Android-застосунку, так як програма побудована саме на взаємодії з користувачем та його даними. Тому більшість функцій в застосунку не будуть працювати без акаунта і, власне, користувачу не буде доступно майже весь функціонал.

Коротка форма:

Use Case: Зареєструватися в системі як користувач.

Актор: Користувач.

Приклад: Користувач відкриває застосунок та бачить Splash Screen (це графічний елемент керування, що складається з вікна, що містить зображення,

логотип і поточну версію ПЗ) [3]. На вибір дається SignUp через пошту або через Google:

1) користувач обирає SignUp через пошту: вводить пошту та пароль і натискає кнопку «Next» та має створений новий профіль;

2) користувач обирає SignUp через Google: відкривається нове вікно, де користувач обирає профіль Google та після має створенний новий профіль.

Поверхнева форма:

Use Case: Зареєструватися в системі як користувач.

Суть: Користувач повинен створити або зайти в свій акаунт у Android-застосунку вивчення сурдоалфавіту.

Передумови: Користувач завантажив застосунок з Google Play.

Основний успішний сценарій:

1) користувач відкриває програму та погоджується із усіма умовами на Splash Screen;

2) користувач переходить на екран «Sign Up»;

3) користувач обирає зручний варіант авторизації: «Sign up with Email» або «Sign up with Google»;

4) якщо користувач обрав «Sign up with Email», то він вводить пошту та пароль, підтверджує і бачить перед собою головний екран застосунку;

5) якщо користувач обрав «Sign up with Google», то він обирає пошту, проходить валідацію від Google, підтверджує та бачить перед собою головний екран застосунку.

Альтернативні сценарії:

Сценарій 1 – Користувач не завантажив застосунок з Google Play. В такому випадку він повинен його завантажити.

Сценарій 2 – Користувач не погодився із умовами використання застосунку та поширення його персональних даних. В такому випадку буде відображатися помилка.

Сценарій 3 – Користувач обрав «Sign up with Email» та не вів або пошту, або пароль. В такому випадку буде помилка, що користувач повинен ввести пошту або пароль.

Сценарій 4 – Користувач обрав «Sign up with Email» та вів не коректну пошту або пароль. В такому випадку буде помилка, що користувач не пройшов валідацію по пошті або паролю.

Сценарій 5 – Користувач не ввімкнув інтернет та образ «Sign Up with Google». В такому випадку буде помилка, про те що немає Інтернет-з’єднання.

Повна форма:

Таблиця 2.1 – Повна форма Usecase

Usecase section	Comment
Use Case Name	Створення профілю користувача
Scope	System
Level	User-goal
Primary Actor	Користувач
Stakeholders and interest	1) користувач: хоче створити профіль у Android-застосунку вивчення сурдоалфавіту, щоб вивчити сурдоалфавіт; 2) Android-застосунок вивчення сурдоалфавіту: підвищення клієнтури, надання можливості користувачам вивчити сурдоалфавіт.
Preconditions	1) користувач не має створеного профілю в Android-застосунку; 2) користувач має акаунт у застосунку.
Success guarantee	Після успішного завершення Use Case у Android-застосунку вивчення сурдоалфавіту буде створений профіль користувача

Main Success Scenario	1) користувач відкриває застосунок вивчення сурдоалфавіту; 2) перед користувачем відкривається Splash Screen; 3) користувач обирає форму реєстрації; 4) користувач обрав реєстрації через пошту: заповнює форму для створення профілю. (або користувач обирає Sign Up With Google); 5) користувач натискає кнопку "Sign Up"; 6) Android-застосунок перевіряє, чи всі обов'язкові поля в формі для створення профілю були заповнені; 7) якщо всі обов'язкові поля були заповнені, Android-застосунок перевіряє, чи дані, введені в форму для створення профілю, є вірними; 8) якщо дані, введені в форму для створення профілю, є вірними, Android-застосунок створює новий профіль; 9) Android-застосунок зберігає дані профілю в БД; 10) користувача перенаправляє на головну сторінку застосунку.
Extensions	1) якщо користувач не заповнив усі обов'язкові поля в формі для створення профілю або вводить не вірні дані, застосунок виведе помилку; 2) якщо користувач вже має створений профіль у Android-застосунку, застосунок виведе повідомлення про те, що профіль вже існує;

	3) якщо користувач не має акаунту у Android-застосунку, застосунок виведе повідомлення про те, що необхідно створити акаунт.
Special Requirements	1) застосунок повинен забезпечувати безпечне зберігання даних профілю; 2) застосунок повинен забезпечувати можливість створення профілю користувача різними мовами.
Technology and Data Variations List	1) застосунок може використовувати технологію штучного інтелекту для аналізу жесту під час проходження тестування на певному уроці; 2) застосунок може використовувати дані з соцмереж (контактів телефону), щоб отримати більш повну картину про користувача.
Frequency of Occurrence	Usecase може виконуватися один раз або кілька разів на день.
Miscellaneous	1) застосунок повинен надавати можливість користувачу редагувати або видаляти свій профіль; 2) застосунок повинен надавати можливість користувачу додавати або видаляти інформацію в свій профіль.

Після створення трьох форм (короткої, поверхневої та повної) usecase для авторизації користувача, можна зробити висновок, що створення акаунту грає ключову роль для коректної роботи Android-застосунку.

2.2 Створення діаграми прецедентів

Діаграма прецедентів [4] – це важливий інструмент для визначення та візуалізації взаємодії між різними аспектами системи та її користувачами. Вона становить ключовий етап аналізу вимог і дозволяє розробникам та зацікавленим

сторонам зрозуміти, як система повинна функціонувати з точки зору користувачів. Ця діаграма складається з набору прецедентів, які представляють сценарії використання системи, та акторів, які її використовують. Актори можуть бути реальними користувачами системи, іншими системами або навіть зовнішніми процесами. Кожен прецедент описує конкретну дію або послугу, яку система може надати своїм акторам. У діаграмі прецедентів актори зображаються паличками або графічними символами, а прецеденти – еліпсами чи колами. Взаємодія між акторами та прецедентами показується за допомогою стрілок. Це дозволяє ясно відображати, які дії можуть бути виконані кожним актором та які прецеденти можуть бути викликані з кожного випадку. Однією з ключових переваг діаграм прецедентів є їх здатність узагальнити велику кількість інформації в лаконічному та зрозумілому форматі. Вони дозволяють виявити потреби користувачів, визначити основні функціональні можливості системи та виявити можливі точки розширення для майбутнього розвитку. Загалом, діаграма прецедентів є потужним інструментом для аналізу та проєктування систем, який допомагає забезпечити високу якість та зручність використання створеної системи для її користувачів. Розглянемо загальну діаграму прецедентів для Android-застосунку вивчення сурдоалфавіту (рис. 2.1).

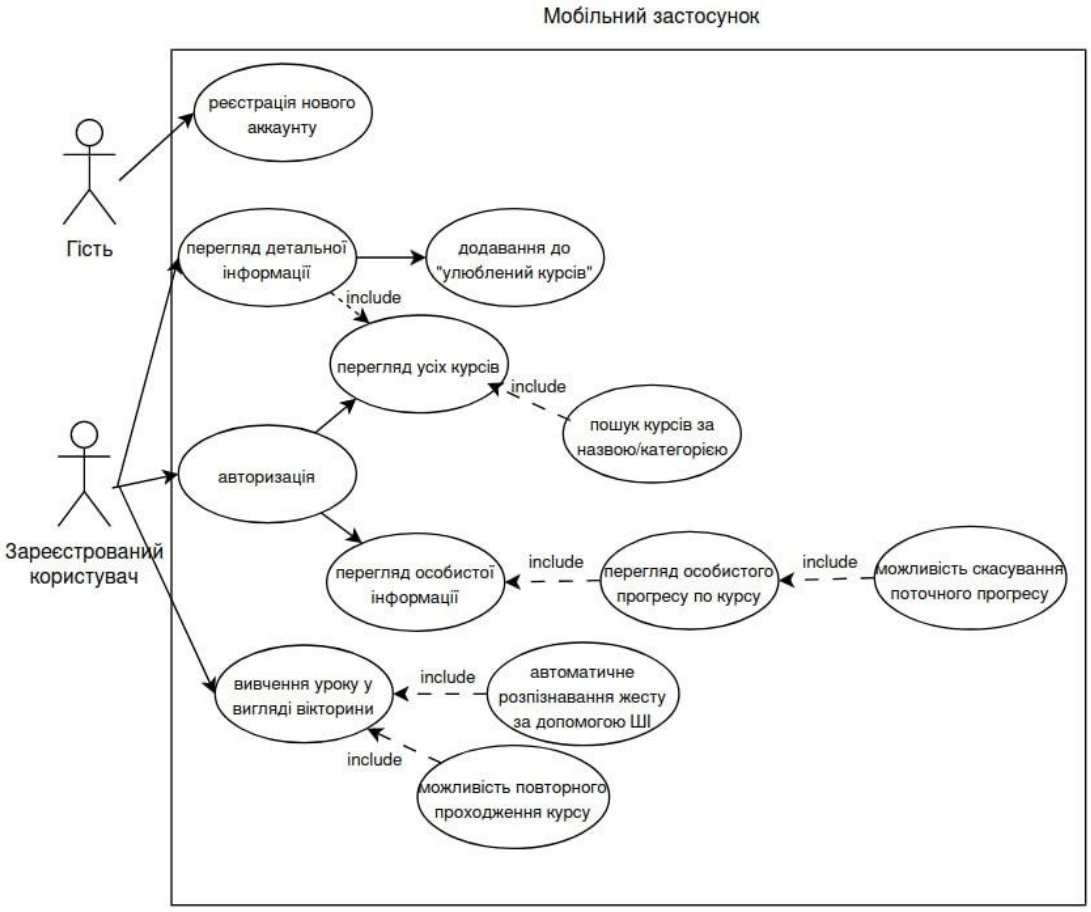

Рисунок 2.1 – Діаграма прецедентів

На діаграмі прецедентів видно, що застосунок може мати лише 2 актори: зареєстрованого та не зареєстрованого користувача, у останнього якого дуже обмежений фунціонал: тільки реєстрація або логін. Зареєстрований користувач має усі доступні функції застосунку, такі як:

1) перегляд усіх курсів із вивчення сурдоалфавіту;
2) додавання курсу до улюблених;
3) перегляд детальної інформації по курсу або уроку;
4) пошук курсів за назвою або іншими параметрами;
5) перегляд особистої інформації на екрані «My Account»;
6) перегляд особистого прогресу по курсу або уроку;
7) можливість скасування поточного прогресу по курсу;
8) вивчення уроку у вигляді вікторини, яка містить технології розпізнавання жестів за допомогою ШІ;

9) можливість повторного проходження курсу.

2.3 Діаграми послідовності

Діаграма послідовності [9] – це не лише інструмент візуалізації взаємодії об'єктів у системі, але і інструмент для аналізу та вдосконалення архітектури ПЗ. Однією з ключових переваг діаграми послідовності є її здатність відобразити різні сценарії взаємодії між об'єктами та процесами в системі. Це дозволяє розробникам легко розуміти, як система повинна працювати в різних умовах та які взаємодії між об'єктами можуть виникати в різних ситуаціях. Діаграми послідовності також можуть бути використані для виявлення потенційних проблем або аномалій у взаємодії між об'єктами. Шляхом аналізу послідовності подій можна виявити можливі конфлікти, перекриття або неочікувані взаємодії, які потребують уваги та вирішення. Крім того, діаграми послідовності допомагають підвищити якість та ефективність комунікації між розробниками, архітекторами та іншими учасниками проєкту. Вони надають графічне зображення взаємодії системи, що сприяє кращому розумінню та уникненню недорозумінь. Таким чином, діаграми послідовності – це не лише інструмент для моделювання взаємодії, але і потужний інструмент для аналізу, комунікації та вдосконалення програмного забезпечення. Розглянемо діаграми послідовності для окремих випадків, наприклад «Відображення списку курсів» (рис. 2.2), «Відображення детальної інформації про курс» (рис. 2.3), «Додавання курсу до розділу улюблених» (рис. 2.4).

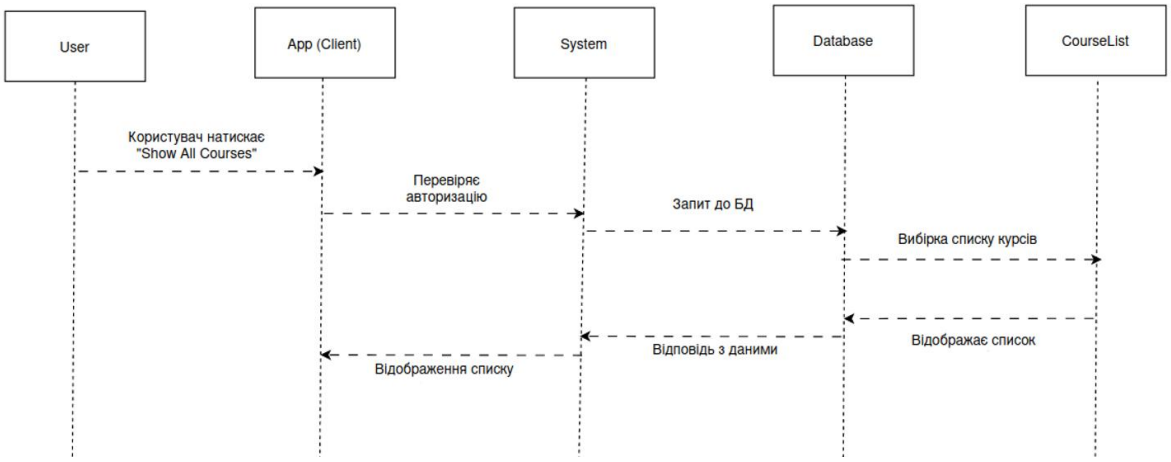

Рисунок 2.2 – Sequence Diagram «Відображення списку курсів»

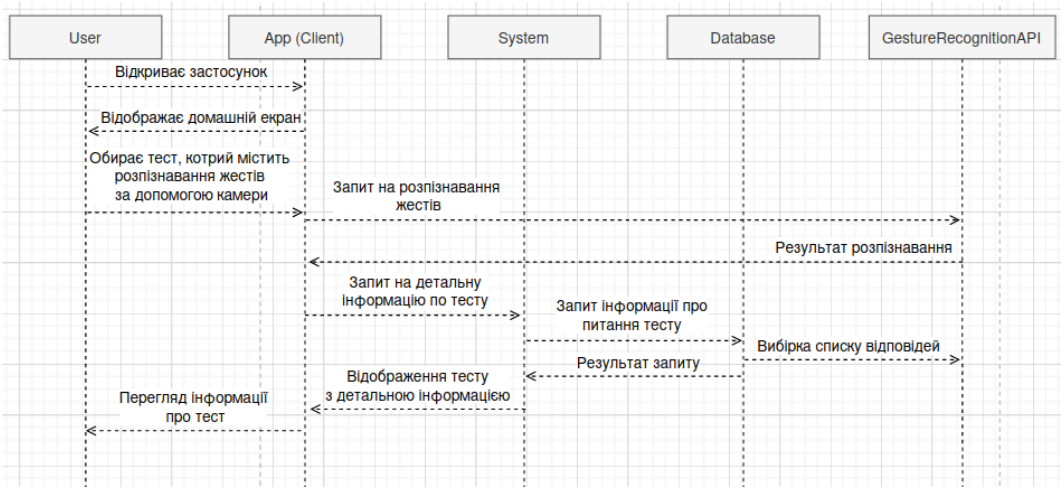

Рисунок 2.3 – Sequence Diagram «Відображення детальної інформації про тест»

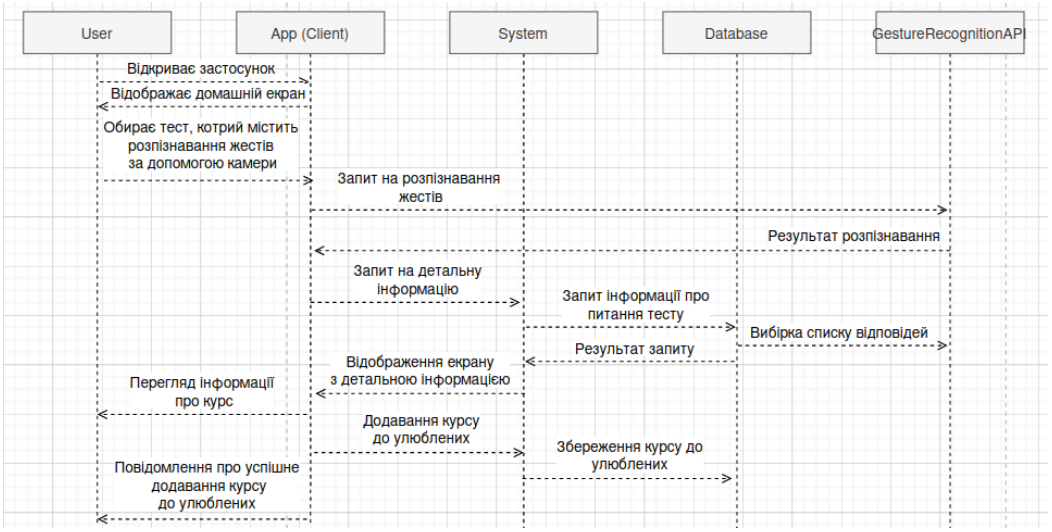

Рисунок 2.4 – Sequence Diagram «Додавання курсу до розділу улюблених»

Діаграма послідовності демонструє взаємодію між об'єктами в системі у визначеній послідовності. Вона дозволяє візуалізувати порядок передачі повідомлень між об'єктами у межах конкретного сценарію використання.

2.4 Діаграми діяльності

Діаграма діяльності [10] є графічним інструментом для моделювання послідовності дій у системі. Цей граф, який схожий на граф станів скінченного автомата, складається з вершин, що представляють конкретні дії, і переходів між ними, які відбуваються після завершення дій.

Дії є основною одиницею специфікації поведінки системи. Кожна дія приймає вхідні сигнали і генерує вихідні. Вони можуть мати пусті вхідні або вихідні множини, або обидві. Виконання дії означає виконання конкретної задачі. Аналогічно, виконання діяльності означає виконання серії дій, які вона включає. Дії можуть виконуватись один або декілька разів під час виконання діяльності. Дії зазвичай обробляють дані, трансформують їх та можуть вимагати певної послідовності.

Діаграми активностей складаються з обмеженої кількості фігур, які з'єднані стрілками. Найважливіші фігури включають скруглені прямокутники для позначення дій, ромби для умовних рішень, риски для позначення паралельних активностей, чорний кружок для позначення початку та кінця процесу, і стрілки, які показують порядок виконання активностей. Діаграма активностей часто порівнюється з блок-схемою в своїй структурі та цілі. Розглянемо детально перші базові діаграми діяльності для двох usecases: «Зміна фото користувача в MyAccount» (рис. 2.5) та «Фільтрація всіх курсів» (рис. 2.6).

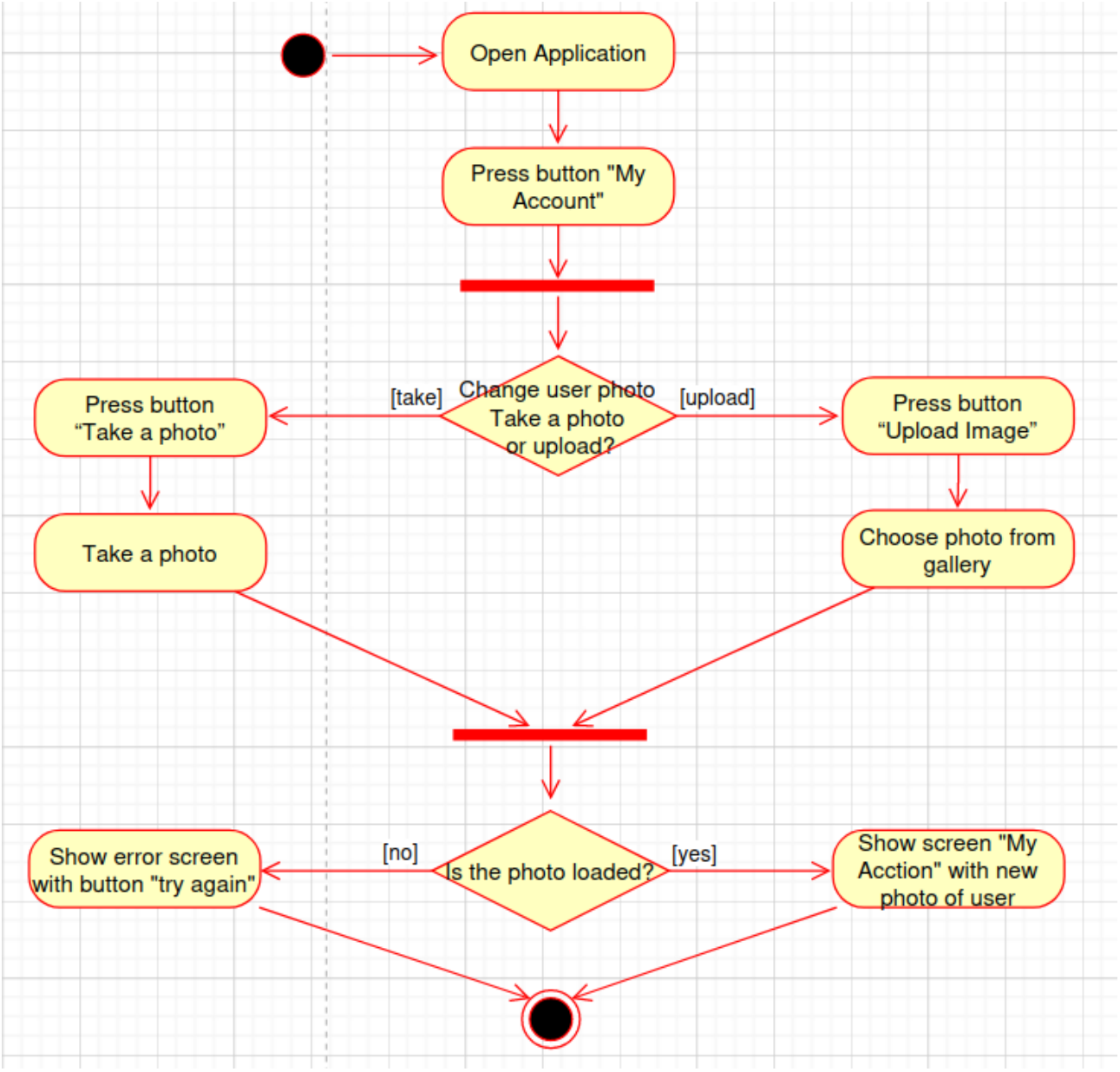

Рисунок 2.5 – Діаграма діяльності для usecase «Зміна фото користувача в My Account»

Ціль: зробити або оновити фото користувача та відобразити його на екрані «My Account».

Початкова точка: відкриття Android-застосунку вивчення сурдоалфавіту.

Кроки:

1) користувач відкриває застосунок;

2) користувач натискає кнопку «My Account», щоб перейти в розділ свого профілю;

3) користувачеві надається вибір: зробити нове фото або завантажити існуюче;

4) функція «Зробити нове фото»: користувач натискає кнопку «Take a photo», щоб відкрити камеру пристрою, робить фото. Система перевіряє, чи завантажено фото і має два стани результату, а саме: **помилка** – якщо фото не завантажено, з'являється екран помилки з кнопкою «Try again» та користувач може натиснути цю кнопку, щоб спробувати зробити фото ще раз. **Успішне завантаження** – якщо фото завантажено, воно з'являється на екрані «My Account» з новим фото користувача;

5) функція «Завантаження фото»: користувач натискає кнопку "Download a picture", щоб вибрати фото з галереї пристрою, обирає фото. Система перевіряє, чи завантажене фото і має два стани результату, а саме: **помилка** – якщо фото не завантажено, з'являється екран помилки з кнопкою «Try again» та користувач може натиснути цю кнопку, щоб спробувати зробити фото ще раз. **Успішне завантаження** – якщо фото завантажено, воно з'являється на екрані «My Account» з новим фото користувача.

Кінцева точка: Відображення екрану «My Account» із новим фото користувача.

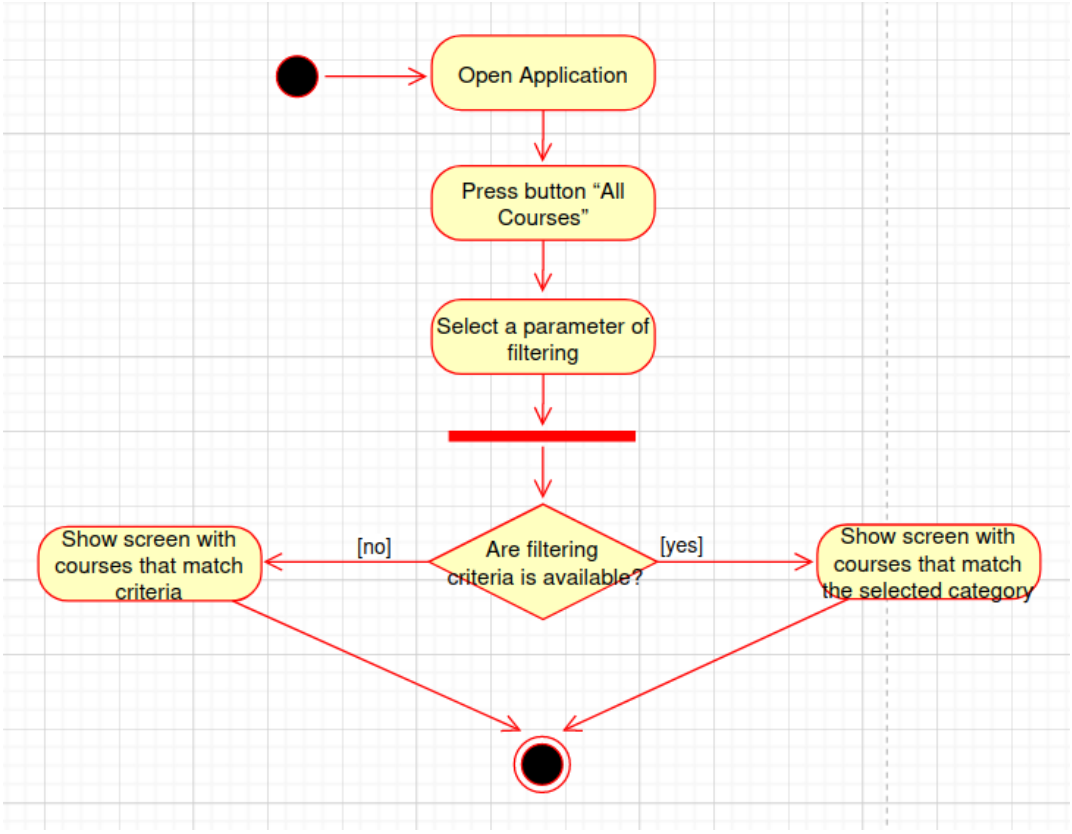

Рисунок 2.6 – Діаграма діяльності для usecase «Фільтрація всіх курсів»

Ціль: відкрити Android-застосунок та вибрати категорію курсів для вивчення сурдоалфавіту.

Початкова точка: відкриття Android-застосунку вивчення сурдоалфавіту.

Кроки:

1) користувач відкриває застосунок;

2) користувач натискає кнопку «All Courses», щоб перейти на сторінку зі списком доступних курсів;

3) вибір категорії курсів: На екрані відображається список категорій курсів, наприклад: легкий чи важкий курс, чи містить курс ШІ для швидкого вивчення сурдоалфавіту;

4) користувач натискає на вибрану категорію курсів;

5) система фільтрує всі курси за обраними параметрами і виводить на екран.

Висновки до розділу 2

У другому розділі кваліфікаційної бакалаврської роботи було змодельовано та детально описано ПЗ, що розробляється у вигляді: діаграми прецедентів, діаграми розгортання, діаграма послідованостей та діаграма діяльності. Завдяки побудові діаграм та написання use case закріплено навички із цієї сфера. У майбутньому це допоможе для підтримка даного ПЗ та реалізації нового функціоналу саме у Android-застосунку.

Написано унікальний use case до реєстрації користувача. Насамперед, розробка Use Case дала змогу виокремити основні сценарії використання застосунку, включаючи різноманітні взаємодії користувачів з його функціоналом. Це підготувало ґрунт для розробки повних Use Case, які детально описують усі можливі сценарії використання програмного забезпечення.

В алгоритмах роботи застосунку були відображені всі основні кроки взаємодії користувачів з програмою, включаючи такі важливі етапи, як пошук курсів, їх вибір та вивчення уроків в цих курсах. Це дозволило забезпечити максимальну зручність та ефективність користувачам при використанні застосунку.

Використані UML-діаграми в даному контексті чітко відображають структуру та вимоги до застосунку, а також архітектуру програмного комплексу в цілому. Це сприяло створенню продукту, який відповідає всім потребам користувачів і має стабільну та ефективну архітектуру.

3 ПРОЄКТУВАННЯ ANDROID-ЗАСТОСУНКУ

3.1 Діаграма розгортання Android-застосунку

Діаграма розгортання [5] – це графічне зображення, яке використовується для візуалізації архітектури та розташування компонентів програмного забезпечення на фізичних або віртуальних обчислювальних пристроях. У випадку Android-застосунків, ця діаграма вказує, як різні частини програми, такі як активності, сервіси, бази даних тощо, розміщуються на пристроях з операційною системою Android.

На діаграмі розгортання для Android застосунку можуть бути показані такі компоненти як:

1) програмні модулі: Це можуть бути активності (Activities), сервіси (Services), приймачі (Broadcast Receivers), а також фрагменти (Fragments), якщо застосунок використовує підходящу архітектуру;

2) фізичні або віртуальні пристрої: Це можуть бути смартфони, планшети, емулятори або будь-які інші пристрої, на яких запускається застосунок;

3) зв'язки між компонентами: Це показується за допомогою стрілок або ліній, які вказують на те, які компоненти взаємодіють між собою. Наприклад, активність може викликати сервіс для виконання певних завдань;

4) залежності від зовнішніх ресурсів: Якщо застосунок взаємодіє з іншими системами або сервісами, це також може бути відображено на діаграмі, щоб показати ці залежності.

Загалом, діаграма розгортання для застосунку допомагає розробникам та зацікавленим сторонам краще зрозуміти, як програмний продукт взаємодіє з фізичними або віртуальними пристроями на платформі Android. Тож, розглянемо детально діаграму розгортання для Android-застосунку вивчення сурдоалфавіту:

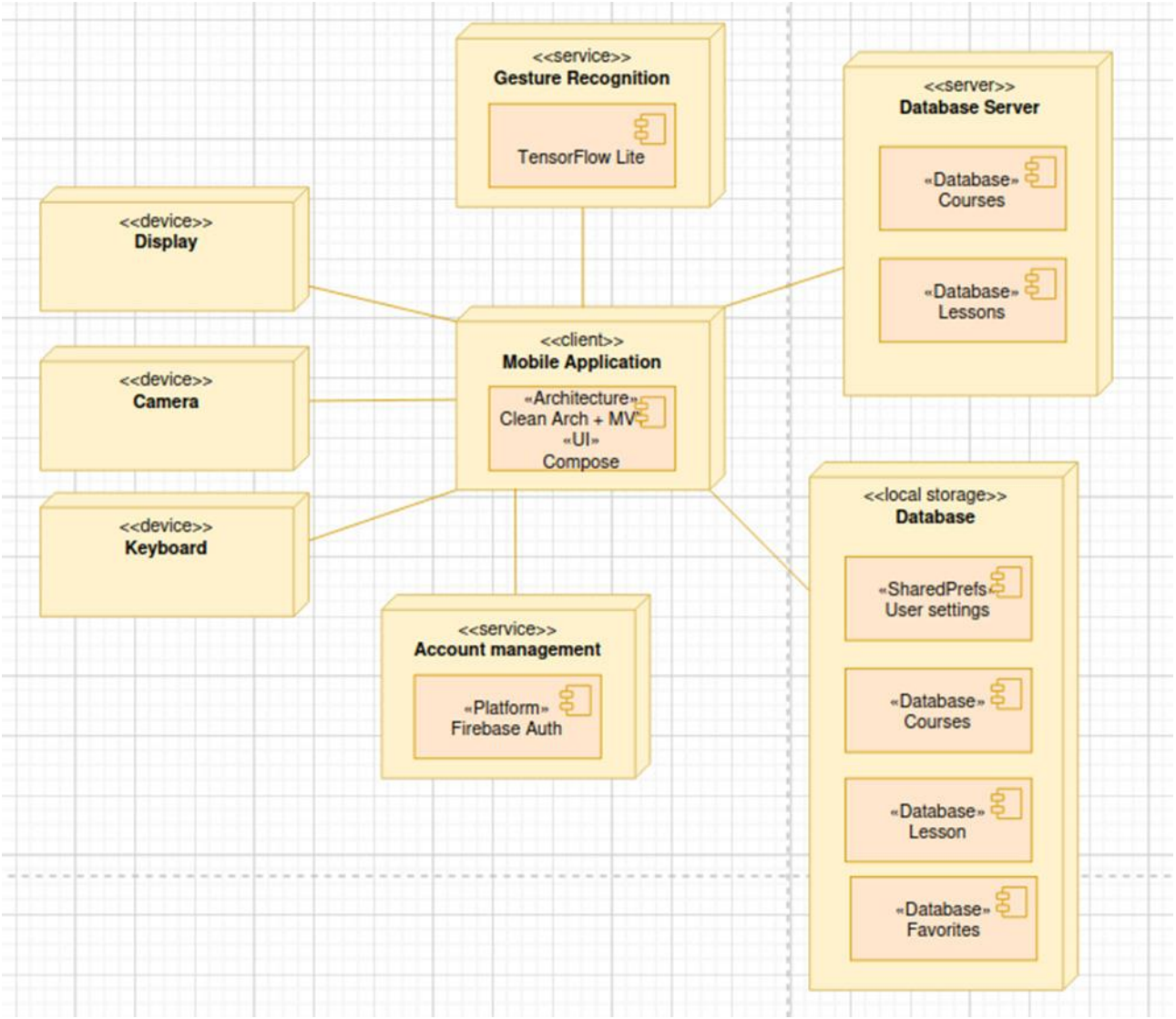

Рисунок 3.1 – Діаграма розгортання

Отже, умовно можна розділити 4 особливі групи для Android-застосунку: Local Storage, Server, Service, Device. Пройдемось по кожному із них.

Local Storage – певна части налаштувань або даних, котрі зберігаються в самому пристрої. Наприклад, користувач має змогу змінити тему застосунку за допомогою екрану «Settings», але після перезапуску застосунку самі ці налаштування мають бути відображені одразу. І зберегти їх можна за допомогою SharedPreferences [6] у Android. Або розглянемо іншу ситуацію, коли користувач використовував застосунок для того, щоб вивчати сурдоалфавіт, але із якихось причин в пристрої зникло Інтернет-з’єднання. Цю проблему можна вирішити за

допомогою локальної БД на самому пристрої, з котрої будуть підтягуватися дані у разі зникнення Інтернет-з’єднання.

Server – на сервері будуть зберігатися питання по курсам та детальний опис інформації по урокам. Реалізовано за допомогою Firebase Realtime Database [7].

Service – містить 2 складові:

1) Gesture Recognition – сервіс, котрий буде використано для розпізнавання зображень, наприклад, бібліотека TensorFlow Lite;

2) Firebase Auth – сервіс, за допомогою якого буде реалізовано реєстрація користувача. Також, є можливість зберігання даних про користувачів, безпеку яких гарантує сам Firebase.

Device – сам пристрій та його складові. В застосунку буде використовуватися тільки: камера (для розпізнавання жестів), клавіатура (наприклад, для пошуку курсів або редагування власної інформації) та дісплей (відображення всього контенту).

Окрім цього, усі ці 4 складові відходять від Client – тобто власне від Android-застосунку та містить в собі:

1) архітектуру – Clean Architecture;
2) архітектурний патерн: MVVM (model-view-viewmodel);
3) бібліотеку для реалізації UI: Jetpack Compose.

3.2 Детальний огляд технологій

Вибір технологій має велике значення, оскільки від нього залежить ефективність роботи ПЗ, а також вимоги до апаратної частини пристрою та його надійність. Для кожного завдання важливо обрати відповідний інструмент для його вирішення. У випадку Android-застосунку вивчення сурдоалфавіту було обрано набір технологій, який широко використовується при розробці мобільного застосунку.

3.2.1 Мова програмування

Kotlin [11, 12] – це мова програмування, яка привертає увагу своєю статичною типізацією та здатністю працювати на платформі JVM та компілюватися в JavaScript. Розроблена командою JetBrains, вона отримала свою назву на честь острова Котлін у Фінській затоці. Основною метою при створенні Kotlin було створення мови, яка була б більш лаконічною та безпечною порівняно з Java, при цьому була б простішою за Scala. Це призвело до збільшення швидкості компіляції та поліпшення підтримки IDE.

Розробка Kotlin розпочалась у 2010 році, а перша публічна версія була представлена в липні 2011 року. Початковий код був відкритий у лютому 2012 року, а вже у червні того ж року була представлена підтримка Android. У грудні 2012 року вийшла версія з підтримкою Java 7. І, нарешті, у листопаді 2015 року була стабілізована основна функціональність мови, готуючи шлях до релізу версії 1.0, який відбувся в лютому 2016 року. З 17 травня 2017 року Kotlin став офіційно підтримуваною мовою для розробки Android-застосунків, а з 7 травня 2019 року був рекомендованим вибором для цієї платформи. На початку листопада 2023 року була випущена мажорна бета-версія Kotlin 2.0.0 Beta1.

Рисунок 3.2 – Логотип Kotlin

Одним з основних переваг Kotlin є те, що вона полегшує деякі обмеження, які існують в Java, наприклад, щодо статичних методів та змінних. Також Kotlin забезпечує сумісність з Java за допомогою анотації JvmName, що дозволяє визначити ім'я класу для використання з Java-проєктами.

3.2.2 Архітектура Clean Architecture Android-застосунків

Clean Architecture (або як її ще називають, чиста архітектура) наголошує на розділенні завдань. Вона не має конкретної кількості шарів, можна додати стільки шарів, скільки потрібно. Основне емпіричне правило полягає в тому, що внутрішній шар не повинен посилатися на зовнішній, і з часом, при руху до центру, деталі повинні ставати абстрактними [13].

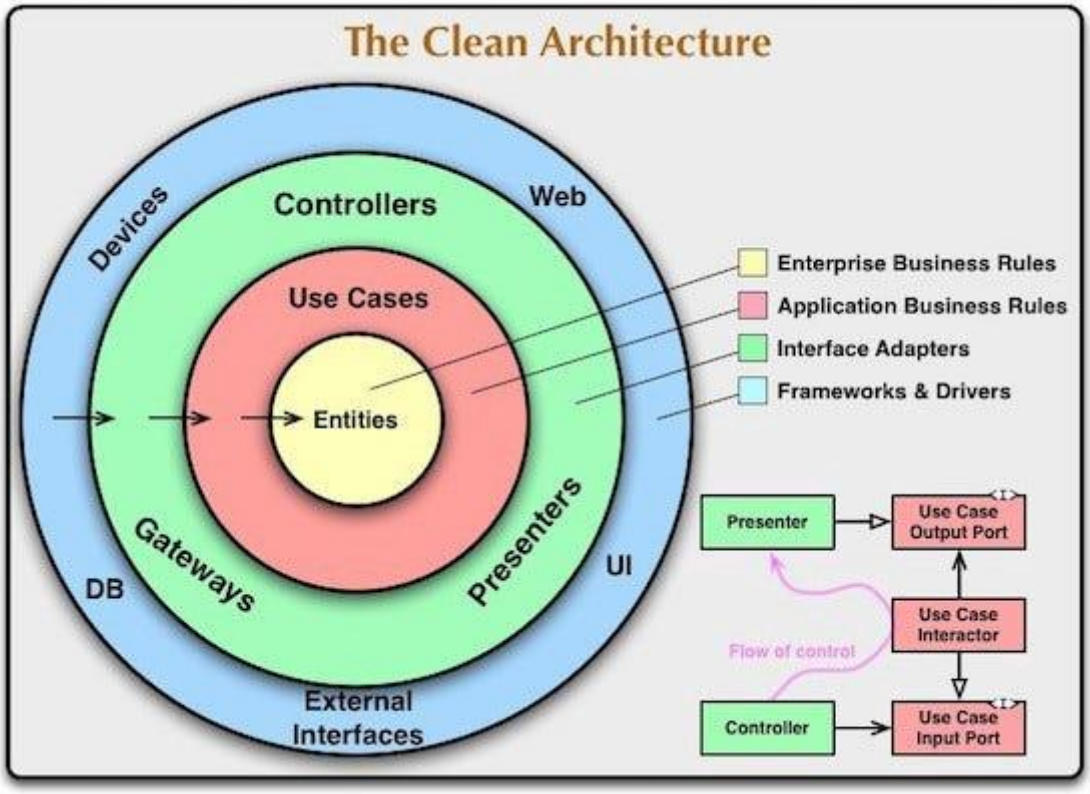

Рисунок 3.3 – Схема чистої архітектури

На рисунку, що зображено вище, зовнішній рівень складається з детальної реалізації, а внутрішній орієнтований на абстракцію. Цей підхід відокремлює бізнес-рівень від низькорівневих залежностей реальних інструментів, які можна використати при розробці ПЗ. Програмне забезпечення повинно бути таким, що зміна інструментів розробки (такого як використаного обладнання та бази даних) не впливає на код, який визначає бізнес-логіку. Чиста архітектура працює на цій загальній ідеї. Можемо уявити шар «Чистої архітектури» як сферу, яка поступово зникає по мірі руху до центру. Центр не має конкретної інформації про те, як

розв'язується та чи інша конкретна задача. Він просто визначає правила або політику, якими має керуватися програмне забезпечення (це бізнес-логіка ПЗ).

3.2.3 Архітектурний патерн model-view-viewmodel

Розробники завжди віддають перевагу чистому та структурованому коду для своїх проєктів. Організація коду за допомогою шаблонів проєктування сприяє підтримці програмного забезпечення. Знання всіх ключових логічних частин Android-застосунку полегшує процес додавання та видалення функцій програми. Крім того, шаблони проєктування також забезпечують покриття всього коду модульним тестуванням без втручання в інші класи. MVVM [14] є визнаним в галузі шаблоном архітектури програмного забезпечення, який подолав недоліки шаблонів проєктування MVP та MVC. MVVM пропонує розділення логіки представлення даних (представлення або користувацький інтерфейс) від основної бізнес-логіки застосунку. Окремі рівні коду у MVVM (рис 3.4): **Модель** відповідає за абстракцію джерел даних. Модель та ViewModel співпрацюють для отримання та збереження даних. **Представлення** – його метою є сповіщення ViewModel про дії користувача. Цей рівень спостерігає за ViewModel та не містить логіки застосунку.

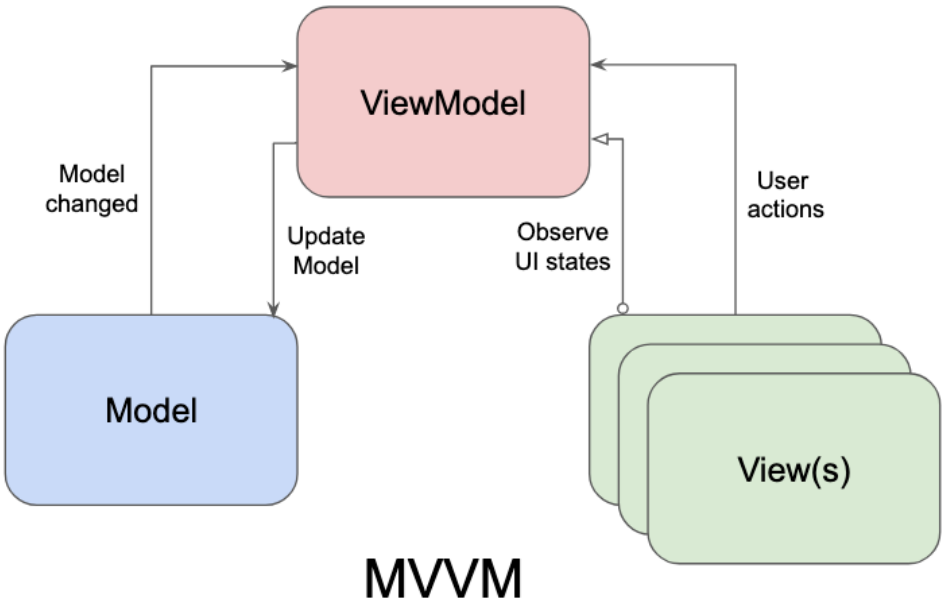

Рисунок 3.4 – Схема чистої архітектури

ViewModel надає потоки даних, пов'язані з представленням (крім того, він служить зв'язковим елементом між Моделлю та Представленням). Шаблон MVVM має деякі схожості з шаблоном проєктування MVP (Model — View — Presenter), оскільки роль Presenter виконує ViewModel. Однак недоліки MVP були вирішені в MVVM, наприклад, ViewModel не містить посилань на Представлення, Існує багато-до-1 відношень між Представленням та ViewModel, Немає методів, які викликають оновлення представлення.

3.2.4 Бібліотеки для Android-застосунку вивчення сурдоалфавіту

Firebase [15] – це інтегрована платформа розробки від Google, яка надає розробникам набір інструментів та сервісів для розробки мобільних та вебзастосунків. Основні можливості Firebase включають в себе аутентифікацію користувачів, зберігання та синхронізацію даних в реальному часі за допомогою бази даних Firestore, зберігання файлів у хмарі, хостинг вебсторінок та застосунків, аналітику використання застосунків, тестування та моніторинг.

Крім цього, Firebase надає можливості для реалізації уведомлень, інтеграції з платежними системами та інші інструменти, що полегшують розробку та впровадження застосунків. Firebase є популярним вибором серед розробників завдяки своїй простоті використання, широкому функціоналу та інтеграції з іншими сервісами Google.

Рисунок 3.5 – Логотип Firebase

Retrofit [16] – це бібліотека для Android, яка дозволяє здійснювати мережеві запити до вебсерверів та обробляти їхні відповіді. Вона базується на стандартних інтерфейсах визначення служб RESTful API і забезпечує простий та зручний спосіб взаємодії з сервером. Однією з основних переваг Retrofit є його декларативний підхід до створення HTTP-запитів: потрібно просто описати структуру запиту за допомогою аннотацій Java або Kotlin на інтерфейсі із визначенням служби, а бібліотека автоматично виконає залишок роботи. Крім того, Retrofit підтримує різні конвертери для серіалізації та десеріалізації даних, такі як JSON, XML або Protobuf, що робить його дуже гнучким для використання з різними типами даних та форматами відповідей сервера.

Room [17] – це архітектурна бібліотека, яка надає абстракцію бази даних SQLite для розробки Android-застосунків. Вона дозволяє розробникам просто та зручно взаємодіяти з базою даних SQLite за допомогою анотованих Java або Kotlin об'єктів даних та SQL-подібних запитів. Room спрощує створення та керування базою даних в Android-застосунках, а також надає рішення для перехідної міграції схеми бази даних. Це дозволяє розробникам швидко та ефективно працювати з локальними даними у своїх застосунках, що робить Room незамінним інструментом для розробки Android-застосунків зі складною логікою даних.

TensorFlow Lite – це оптимізована версія бібліотеки машинного навчання TensorFlow, призначена для виконання на пристроях з обмеженими ресурсами, таких як мобільні телефони, IoT-пристрої, вбудовані системи та інші. Вона надає широкий набір інструментів для створення, оптимізації та виконання моделей машинного навчання навіть на пристроях з обмеженими обчислювальними можливостями і енергоефективною архітектурою [18].

Рисунок 3.6 – Логотип TensorFlow Lite

TensorFlow Lite дозволяє розробникам вбудовувати моделі машинного навчання безпосередньо в їхні застосунки, що дозволяє реалізувати інтелектуальні функції без необхідності постійного з'єднання з хмарними сервісами. Вона підтримує різноманітні типи моделей, включаючи конволюційні нейронні мережі (CNN), рекурентні нейронні мережі (RNN), а також інші алгоритми машинного навчання.

Основні переваги TensorFlow Lite включають в себе високу швидкодію виконання моделей на пристроях з обмеженими ресурсами, малу величину файлів моделей, що дозволяє ефективно використовувати обмежену пам'ять пристрою, а також підтримку широкого спектру апаратних платформ, включаючи CPU, GPU та спеціалізовані прискорювачі машинного навчання.

Модель TensorFlow Lite, яка використовується у застосунку Android, вона приймає дані для обробки та генерує прогнози згідно зі своєю логікою. Це може бути, наприклад, класифікація зображень, розпізнавання мовлення або будь-яке

інше завдання машинного навчання. Спеціальне середовище виконання TensorFlow Lite на пристрої Android відповідає за виконання моделі найефективнішим способом для даного пристрою, забезпечуючи швидку та ефективну роботу. Важливо, щоб дані, які надходять до моделі, були у відповідному форматі тензорів, який є основним типом даних для роботи з TensorFlow. Коли модель обробляє вхідні дані, вона генерує результати прогнозу, які можуть бути новими тензорами. Ці результати можуть використовуватися для різних цілей, таких як відображення користувачу, збереження у пам'яті або подальша обробка додаткових дій [18].

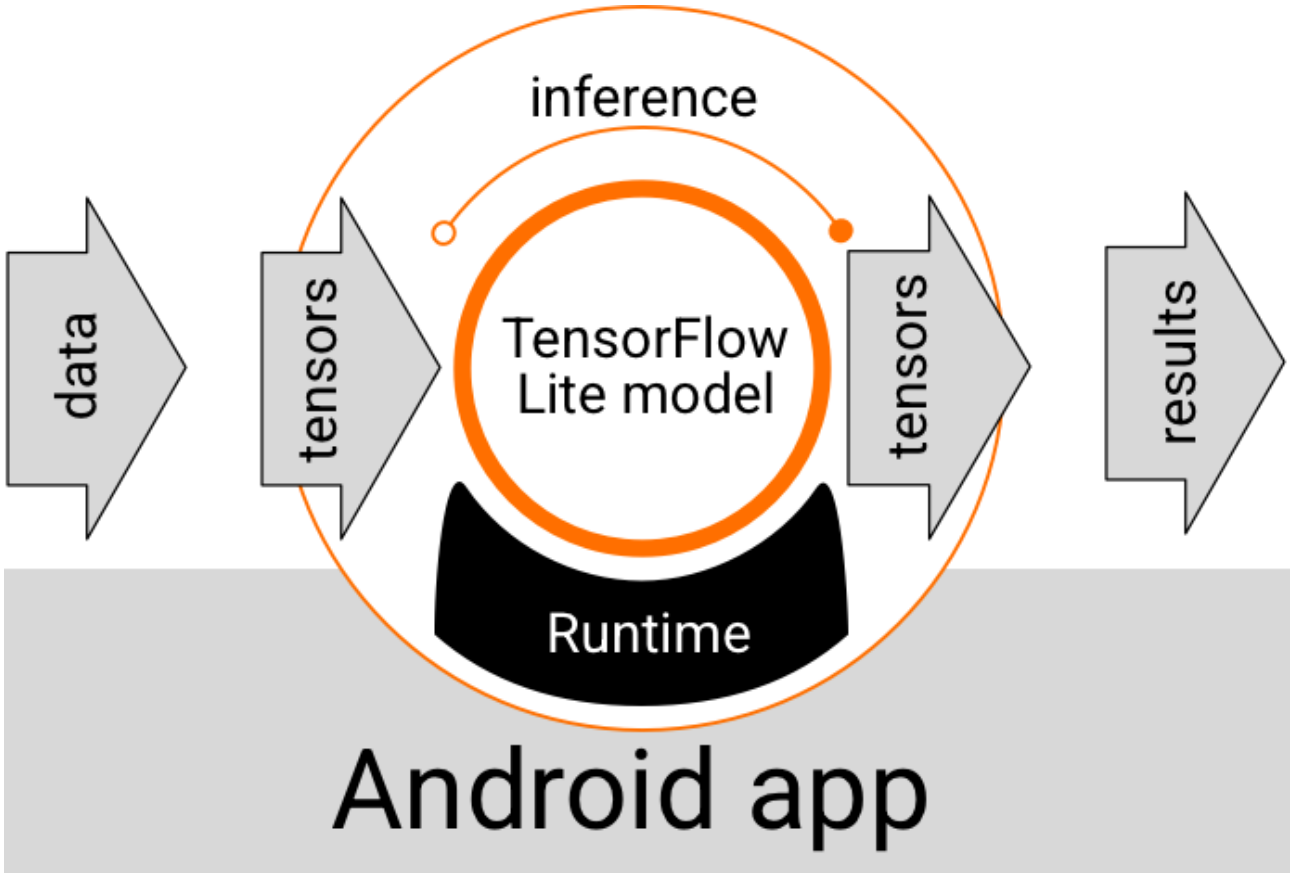

Рисунок 3.7 – Функціональний потік виконання для моделей TensorFlow Lite у Android-застосунках

На рівні функціонального дизайну програми Android для роботи з моделлю TensorFlow Lite необхідні наступні компоненти:

1) середовище виконання моделі TensorFlow Lite, яке забезпечує виконання моделі на пристрої;

2) обробник введення, який перетворює вхідні дані у формат тензорів для моделі;

3) обробник виходу, який отримує результати прогнозу та інтерпретує їх для подальшого використання в програмі.

Для реалізації функції розпізнавання жестів з використанням ШІ потрібно навчити модель і зробити це можна за допомогою Google Teachable Machines [22]. Це вебінструмент (рис. 3.8), який спрощує створення моделей ML. Він дозволяє людям без досвіду програмування створювати моделі, які можуть розпізнавати зображення, звуки та пози.

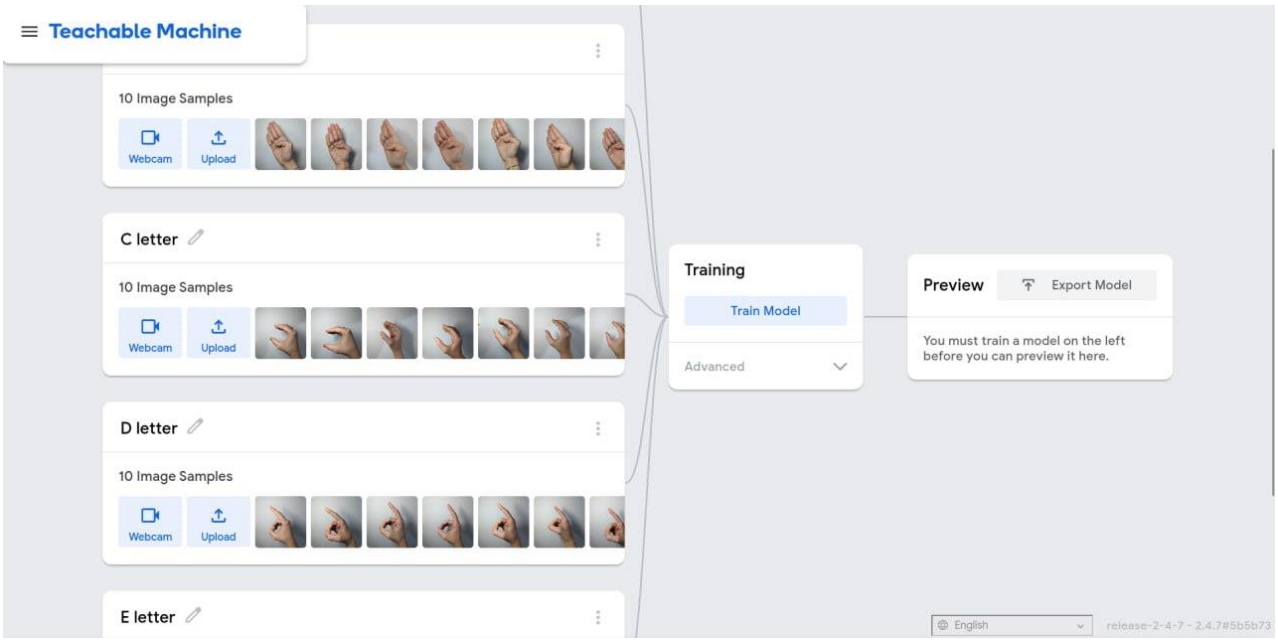

Рисунок 3.8 – UI вебсайту Google Teachable Machines

Ключові особливості Google Teachable Machines:

– простота використання: є можливість завантажувати зображення, записувати звуки або використовувати камеру для навчання вашої моделі;

– універсальність: Teachable Machines може використовуватися для різних проєктів, таких як створення вебсайтів, програм та інтерактивних пристроїв;

– конфіденційність: є можливість тренувати модель локально на своєму пристрої без передачі даних Google.

3.3 Діаграма класів

Діаграма класів [19] є важливим інструментом у процесі розробки програмного забезпечення. Вона допомагає розробникам візуалізувати структуру системи та взаємозв'язки між її компонентами. Завдяки діаграмам класів можна легко розуміти, як об'єкти взаємодіють один з одним, які атрибути та методи вони мають, а також які зв'язки між класами існують. Під час проєктування системи розробники створюють діаграми класів для кожного компонента програми. Крім того, діаграми класів можуть бути використані для документування коду, що полегшує розуміння системи для інших членів команди або сторонніх розробників. Один з основних принципів діаграм класів - це модульність. Кожен клас повинен мати чітко визначену функціональність і відповідати лише за свою частину роботи. Діаграми класів також можуть бути використані для виявлення можливих проблем архітектури програми, таких як надмірне зв'язування або недостатня зрозумілість логіки програми. Вони дозволяють розробникам зосередитися на проєктуванні ефективних та гнучких систем, що можуть легко масштабуватися та розвиватися у майбутньому.

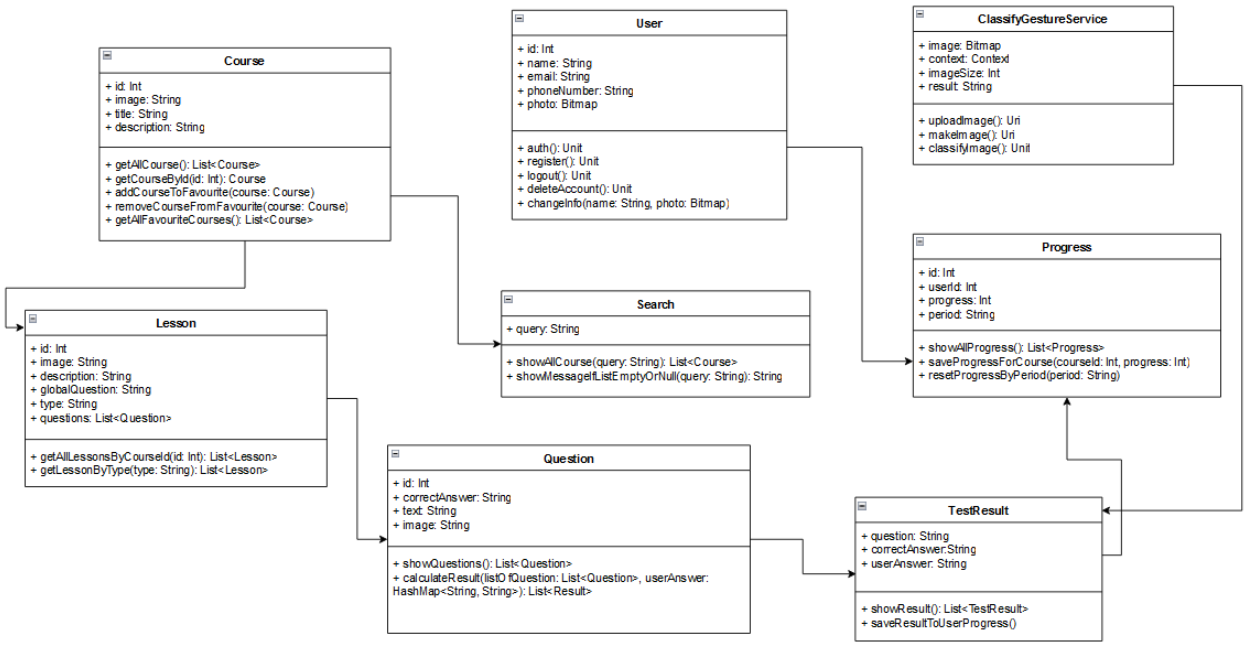

Рисунок 3.9 – Діаграма класів

Опис основних класів, які представлені на діаграмі класів:

1) Course – клас, що містить інформацію про курс, а саме: унікальний ідентифікатор, зображення, назву та опис. Має методи отримання всіх курсів, отримання курсу по унікальному ідентифікатору, додавання чи видалення курсу з списку улюблених або отримання всіх улюблених курсів;

2) Lesson – це клас, котрий описує структуру уроку у курсі. Має поля, такі як унікальний ідентифікатор, зображення, опис, глобальне запитання, тип та список питань. Методи дозволяють отримати усі уроки за певним типом або унікальним ідентифікатором;

3) Question – представляє запитання у певному типі уроку. Містить в собі унікальний ідентифікатор, зображення, текст та правильну відповідь. Є можливість показувати усі уроки та в кінці тесту підрахувати кількість правильних відповідей;

4) TestResult – клас, який містить в собі розрахунки результату після тесту. Має всю необхідну інформацію: питання, правильну відповідь та відповідь користувача. За допомогою цього класу, можна показати весь результат користувачеві та зберегти його до прогресу;

5) Progress – клас, який відповідає за прогрес користувача за певний проміжок часу, тому, власне, і має такі поля як: унікальний ідентифікатор, ідентифікатор користувача, прогрес та період. Можна показати увесь прогрес, зберегти прогрес для певного курсу або прибрати прогрес за певний період;

6) Search – допомогає зручно фільтрувати усі курси за певним запитом і має два стани: або показати усі курси, або показати помилку, що курсів за таким запитом немає;

7) ClassifyGestureService – сервіс, котрий допомогає при певному типі уроку підрахувати результат за допомогою ШІ. Має такі поля, як: зображення жесту, контекст застосунку (для того щоб можна було використовувати класи системи Android), розмір зображення та результат. Містить прості методи, такі

як зробити зображення через камеру або взяти з галереї та, власне, розпізнати саме зображення;

8) User – клас, який містить опис користувачів та певні маніпуляції із їхніми даними, такі як: зміна персональної інформації, створення або видалення нового акаунту, вихід з поточної сесії та інше.

Таким чином, можна зробити висновок, що діаграма класів – це необхідний інструмент для кращого розуміння ПЗ, що розробляється.

3.4 Модель локальної БД

Модель бази даних [20], також відома як схема бази даних, є описом структури та організації даних у конкретній системі управління базами даних (СУБД). Вона визначає, як саме дані будуть зберігатися на фізичному носії, які типи даних будуть використовуватися, які ключі та індекси будуть створені, а також які обмеження цілісності будуть застосовуватися. Модель бази даних розробляється на основі моделі даних, яка є більш абстрактним описом організації та взаємозв'язків даних у системі.

Елементи моделі бази даних:

– **таблиці** є основними компонентами для зберігання даних у базі даних. Вони складаються з рядків і стовпців, де кожен рядок представляє окремий запис, а кожен стовпець містить атрибут цього запису, можуть містити різні типи даних, такі як числові, текстові, дати та інші;

– кожен **стовпець** у таблиці визначає певну характеристику або атрибут даних. Стовпці мають ім'я, яке повинно бути унікальним у межах таблиці, і тип даних, який визначає, які значення можуть зберігатися в цьому стовпці;

– **рядки** в таблиці представляють окремі записи або екземпляри даних. Кожен рядок складається з набору значень, по одному для кожного стовпця таблиці. Рядки можуть ідентифікуватися за допомогою первинного ключа, що гарантує унікальність кожного запису;

– **ключі** забезпечують унікальну ідентифікацію рядків у таблиці. Первинний ключ – це стовпець або комбінація стовпців, яка однозначно ідентифікує кожен рядок. Зовнішні ключі використовуються для встановлення зв'язків між таблицями;

– **індекс** – це структура даних, яка зіставляє значення стовпців з розташуванням відповідних рядків у таблиці;

– **обмеження цілісності** можуть регулювати типи даних, допустимі значення, зв'язки між таблицями та інші аспекти.

Розглянемо детальніше схему локальної БД на рисунку нижче:

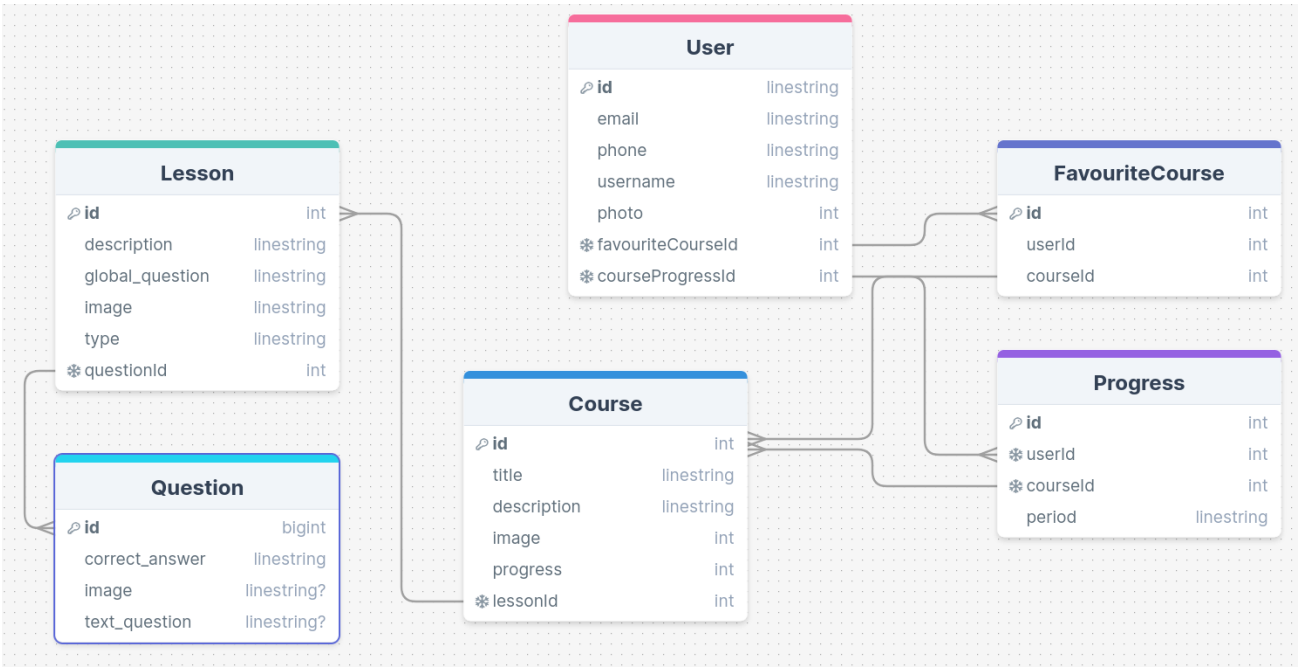

Рисунок 3.10 – Модель БД

Зв’язки між класами:

1) User – FavouriteCourse: one-to-many – користувач може мати багато улюблюненних курсів;

2) Course – Lesson: one-to-many – курс може мати багато уроків;

3) Lesson – Question: one-to-many – урок може мати багато запитань;

4) User – Progress: one-to-many – користувач може мати прогрес по багатьом курсам;

5) Course – Progress: one-to-many – один курс може мати декілька прогресів за певний проміжок часу.

3.5 Діаграма пакетів

Діаграми пакетів [21] в мові моделювання UML служать для відображення залежностей між різними пакетами, що утворюють модель системи. Пакет в UML – це елемент моделі, який використовується для групування інших елементів моделі. Він має власних членів і володіє ними. Якщо пакет видаляється з моделі, усі його члени також видаляються. У діаграмах пакетів в UML можуть бути використані спеціальні відносини залежності:

– імпорт пакету: вказує на те, що простір імен додає назви членів пакету до свого власного простору імен;

– злиття пакету: показує, що вміст двох пакетів має бути об'єднаний, подібно до узагальнення.

Діаграми пакетів можуть використовуватися для різних цілей, таких як відображення функціональності системи за допомогою прецедентів або ілюстрація архітектури програмного комплексу з різними шарами. Вони дозволяють візуалізувати залежності між цими пакетами та механізми їх зв'язку.

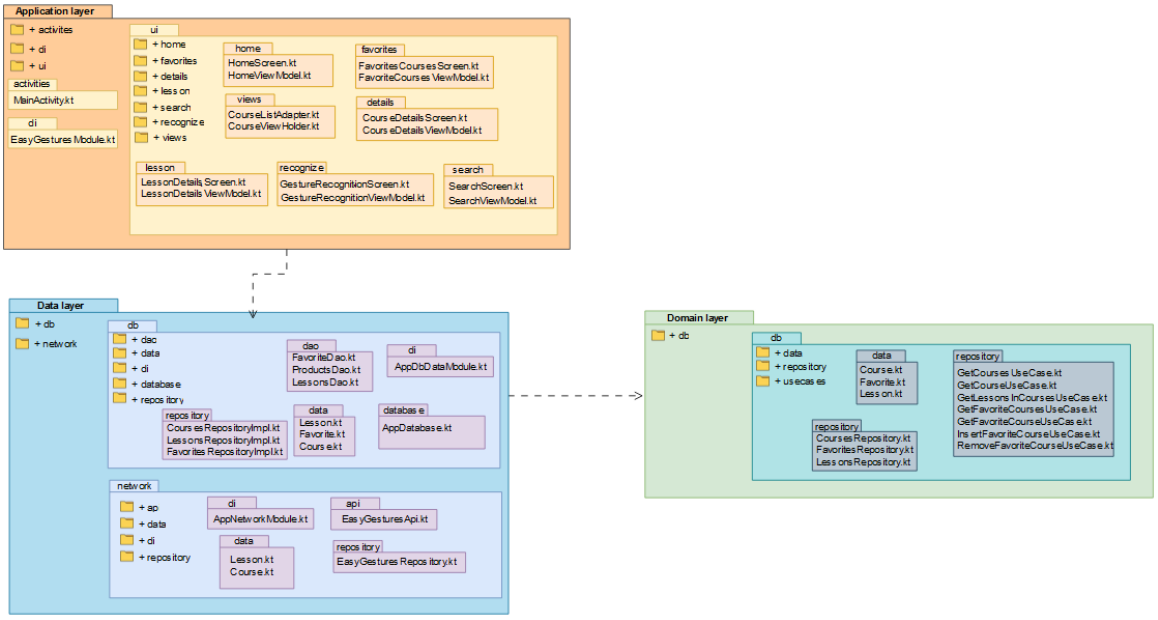

Рисунок 3.11 – Діаграма пакетів

Діаграма пакетів є високорівневим уявленням структури програми. Вона показує, як різні пакети програми пов'язані один з одним.

Application layer містить модулі, які відповідають за функціональність програми. Модуль Activities містить класи, які реалізують інтерфейс програми користувача. Модуль Data містить класи, які працюють із даними. Модуль Network містить класи, які взаємодіють із мережею. Модуль DI містить класи, що використовуються для впровадження залежностей.

Domain layer містить моделі даних, які є сутністю програми.

Data layer містить класи, які використовуються для доступу до даних та Android залежності.

Висновки до розділу 3

В третьому розділі виконана робота з моделювання та конструювання ПЗ, а саме Android-застосунку вивчення сурдоалфавіту. Розроблено діаграму розгортання, де чітко видно, як спроєктована архітектура в мобільному застосунку. Детально описано технології, архітектура, архітектурний патерн та бібліотеки, котрі будуть використовуватися в Android-застосунку та проведено аналіз вказаних технологій. Вказано базову діаграму класів та детальну діаграму пакетів для кращого розуміння ієрархії архітектури застосунку. Закріплено навички із розробки UML-діаграм.

4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ANDROID-ЗАСТОСУНКУ

Мобільний застосунок вивчення сурдоалфавіту з використанням технології жестів містить в собі 6 основних сторінок для авторизованого користувача. Застосунок створенний лише для двух типів користувачів: авторизованого та ні. Відповідно сформованих вимог та фіункціоналу системи, користувач, а саме застосунок користувача, виконує CRUD-операції, API запити задля вивчення сурдоалфавіту.

4.1 Огляд дизайну Android-застосунку

Для неавторизованого користувача доступна тільки «Login Screen» (рис. 4.1), а для авторизованого користувача – сторінка «Home Screen» (рис. 4.4) із базовою інформацією про застосунок, така як: «My Account», «Favourite Course», «Search Course».

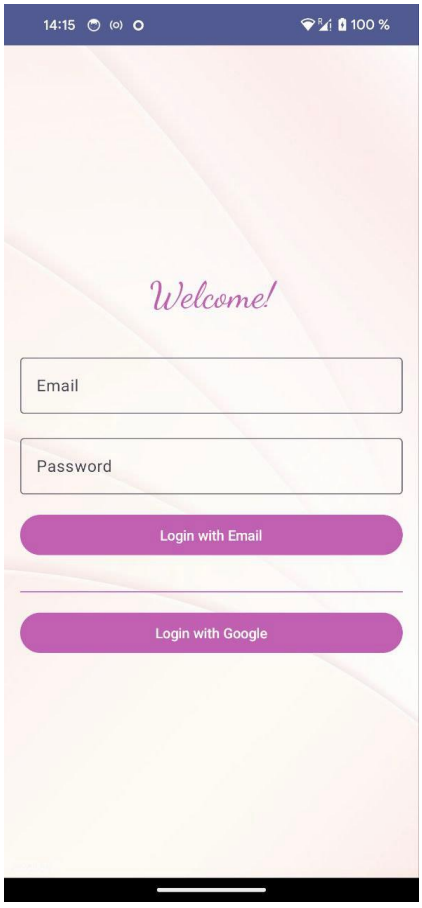

Рисунок 4.1 – Екран авторизації користувача

Логіку авторизації зроблено за допомогою Firebase Auth – він допомогає відстежувати коли користувач створив акаунт в застосунку, яким методом і можна власне побачити інформацію про самого користувача за допомогою комфортної таблиці (рис. 4.2).

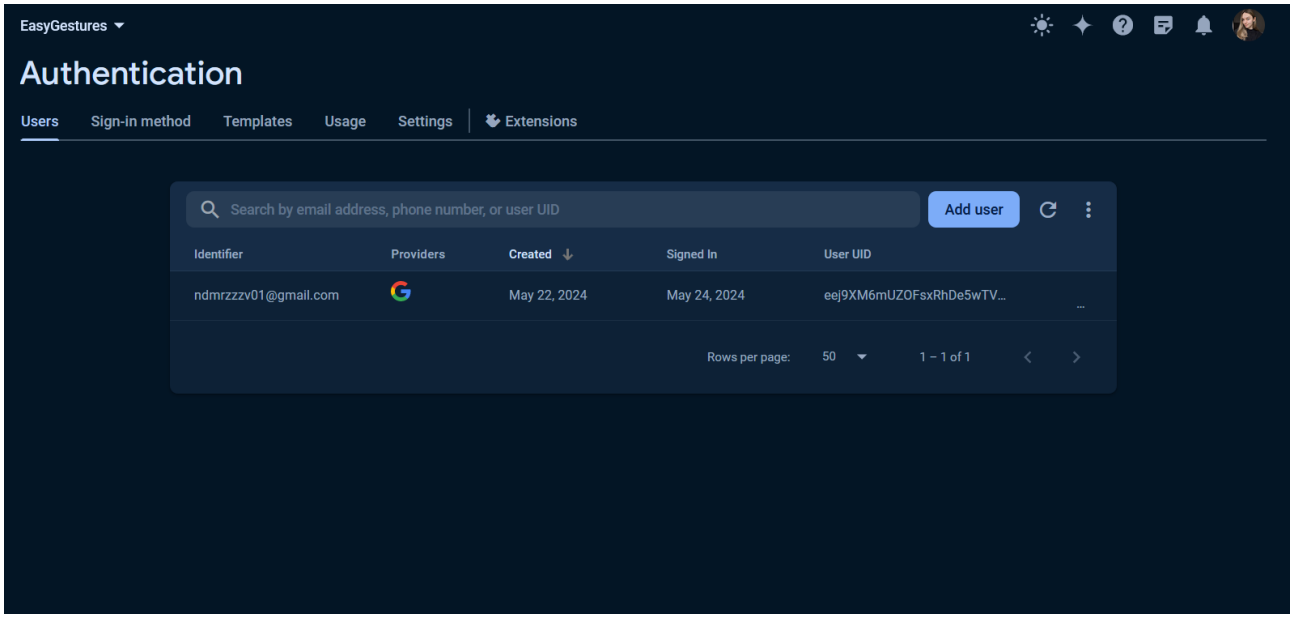

Рисунок 4.2 – Зареєстровані користувачі в застосунку

В Android є бібліотека (рис. 4.3) , котра дозволяє комфортно працювати із Firebase Auth.

```
dependencies {
    // Import the BoM for the Firebase platform
    implementation(platform("com.google.firebase:firebase-bom:33.0.0"))

    // Add the dependency for the Firebase Authentication library
    // When using the BoM, you don't specify versions in Firebase library dependencies
    implementation("com.google.firebase:firebase-auth")
}
```

Рисунок 4.3 – Імпорт бібліотеки

І додаємо легкий код для взаємодії із цією бібліотекою:

```
class LoginViewModel(
        context: Context
) : ViewModel() {

        private val _authState = MutableState-
Flow<AuthState>(AuthState.None)
        val authState: StateFlow<AuthState> = _authState
```

```
        private val auth: FirebaseAuth = FirebaseAuth.getInstance()

        private val gso = GoogleSignInOptions.Builder(GoogleSignInOp-
tions.DEFAULT_SIGN_IN)
            .requestIdToken(context.getString(R.string.default_web_cli-
ent_id))
            .requestEmail()
            .build()

        private val client = GoogleSignIn.getClient(context, gso)

        fun signInWithEmailAndPassword(email: String, password: String)
{
            auth.signInWithEmailAndPassword(email, password)
                .addOnCompleteListener { task ->
                    _authState.value = if (task.isSuccessful) {
                    AuthState.Success
                    } else {
                    AuthState.Error(task.exception?.message ?: "Unknown
error")
                    }
                }
            }

        fun handleGoogleSignInResult(account: GoogleSignInAccount) {
            val credential = GoogleAuthProvider.getCredential(ac-
count.idToken, null)
            auth.signInWithCredential(credential)
                .addOnCompleteListener { task ->
                    _authState.value = if (task.isSuccessful) {
                    AuthState.Success
                    } else {
                    AuthState.Error(task.exception?.message ?: "Unknown
error")
                    }
                }
            }

        fun getGoogleSignInClient() = client

}

sealed class AuthState {
    object None : AuthState()
    object Success : AuthState()
    data class Error(val message: String) : AuthState()
}
```

Після успішної авторизації, як було зазначено вище, користувач побаче «Home Screen» (рис. 4.4).

Рисунок 4.4 – Головна сторінка застосунку

Бачимо три основні сторінки: «My Account» (рис. 4.5), «Favourite», «Search» (рис. 4.6).

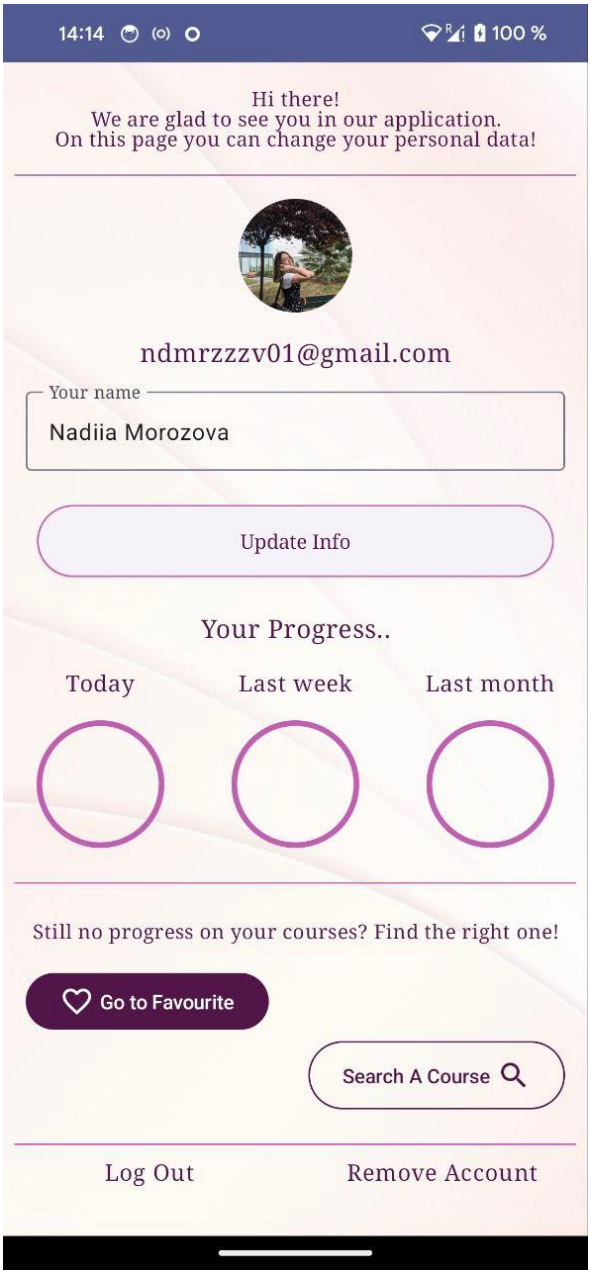

Рисунок 4.5 – Головна сторінка застосунку

На сторінці «My Account» користувач може змінювати персональну інформацію, таку як фото та ім’я, але після натискання кнопки «Update info». Окрім цього, виводиться поточний прогрес по курсам у трьох періодах: сьогодні, минулий тиждень, минулий місяць. На даному скріншоті застосунку, прогрес користувача дорівнює 0, тобто користувач не проходив жодного курсу за будь-який період. У користувача є можливість вийти з поточного акаунту або

видалити його. Зі сторінки «My Account» можна одразу переміститися у дві інші сторінки: «Favourite», «Search» (рис. 4.6).

Рисунок 4.6 – Сторінка пошуку курсів

На сторінці пошуку курсів, якщо користувач не ввів жодного курсу, то на екрані показуються усі можливі курси вивчення алфавіту. Інформація про курс зберігається у Firebase Realtime Database (рис. 4.7).

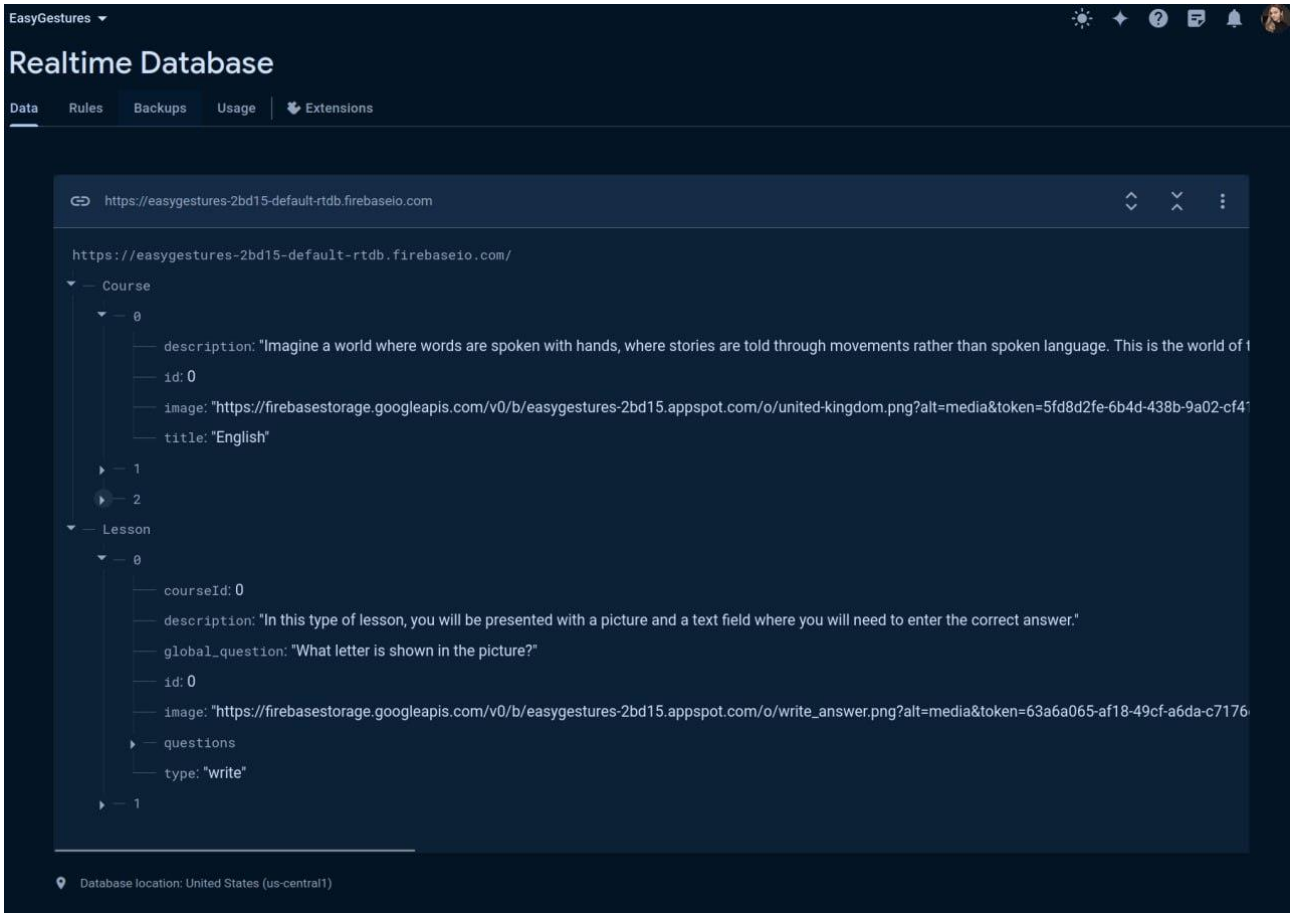

Рисунок 4.7 – Збереження інформації про курси

Firebase Realtime Database – це хмарна база даних NoSQL, що дозволяє зберігати та синхронізувати дані в реальному часі. Бачимо, що курси мають зображення, тобто в цьому випадку, прапор мови, котру можна вивчити. Самі зображення зберігаються у Firebase Storage – це служба зберігання об'єктів, до якої можна отримати доступ через Google Cloud Platform. За допомогою Firebase Storage можна отримувати доступ до файлів за посиланнями, легко завантажувати файли, а також відстежувати хід виконання завдань. На рисунку нижче наведно список усіх можливих файлів проєкту.

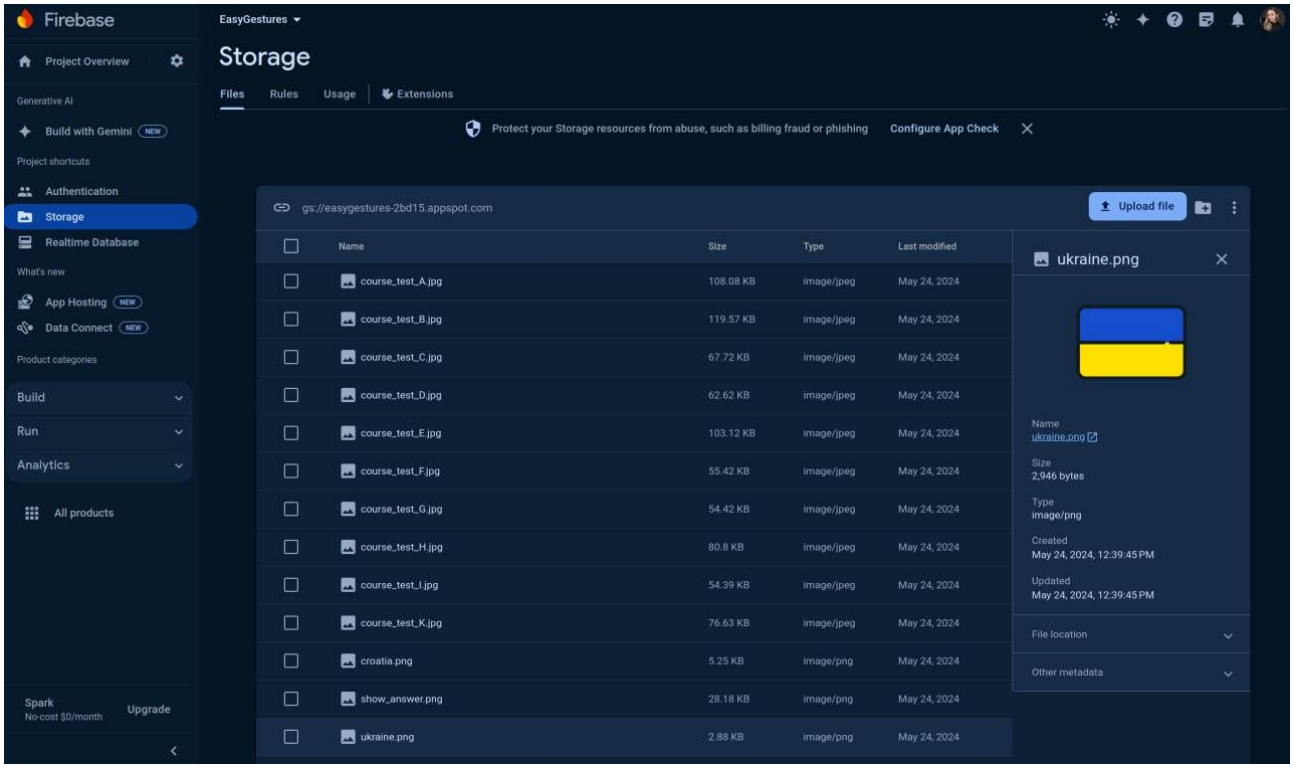

Рисунок 4.8 – Збереження зображень

Власне, після завантаження зображень у Storage та заповнення бази даних Firebase Realtime Database, використовуємо код, котрий зазначений у документації Firebase для запиту інформації про курси. Також, додаємо розширення для класу DatabaseReference від Firebase для комфортної роботи із асинхроними запитами. В нашому випадку, використовуємо Kotlin Coroutines [23] – це шаблон проектування паралелізму, який спрощує виконання асинхронних завдань в Android. Їх додано в Kotlin у версії 1.3 і вони базуються на добре відомих концепціях з інших мов програмування. В Android coroutines допомагають керувати довготривалими завданнями, які можуть заблокувати основний потік і призвести до зависання програми. Більше 50% професійних розробників, які використовують coroutines, відзначили підвищення продуктивності.

```
class CourseNetworkRepositoryImpl(
        private val databaseReference: DatabaseReference
): CourseNetworkRepository {

        override suspend fun getAllCourses(): List<Course> {
```

```
            val listOfCourse = mutableListOf<Course>()
            val references = databaseRefer-
ence.child("Course").awaitSingle()?.children
            if (references != null) {
                for (item in references) {
                    val course = item.getValue<Course>()
                    if (course != null) {
                        listOfCourse.add(course)
                    }
                }
            }
            return listOfCourse
        }
}
```

Та розширення до класу DatabaseReference:

```
suspend fun DatabaseReference.awaitSingle(): DataSnapshot? =
        suspendCoroutine{continuation->
            val listener = object:ValueEventListener {
                override fun onDataChange(snapshot:DataSnapshot){
                    try {
                        continuation.resume(snapshot)
                    } catch(exception:Exception) {
                        Log.d("Exception",exception.message.toString())
                    }
                }
                override fun onCancelled(error:DatabaseError){
                    continuation.resumeWithException(error.toExcep-
tion())
                }
        }
        this.addValueEventListener(listener)
}
```

Після виведення інформації про курси, користувач обирає, наприклад, вивчення англійського сурдоалфавіту і бачить перед собою сторінку «Choose lesson type» (рис. 4.9).

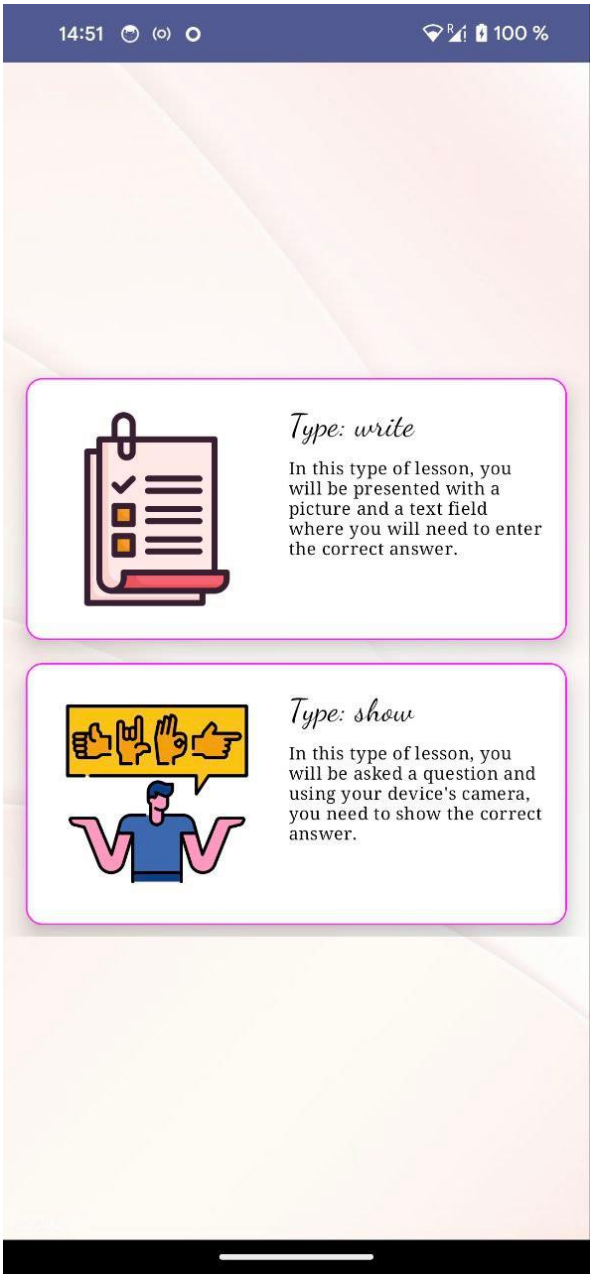

Рисунок 4.9 – Вибір типу уроку

В застосунку існує 2 варіанти уроку:

1) урок, де користувач вводить відповідь письмово (рис. 4.10);

2) урок, де користувачу виводиться питання і він має завантажити зображення жесту (рис. 4.11).

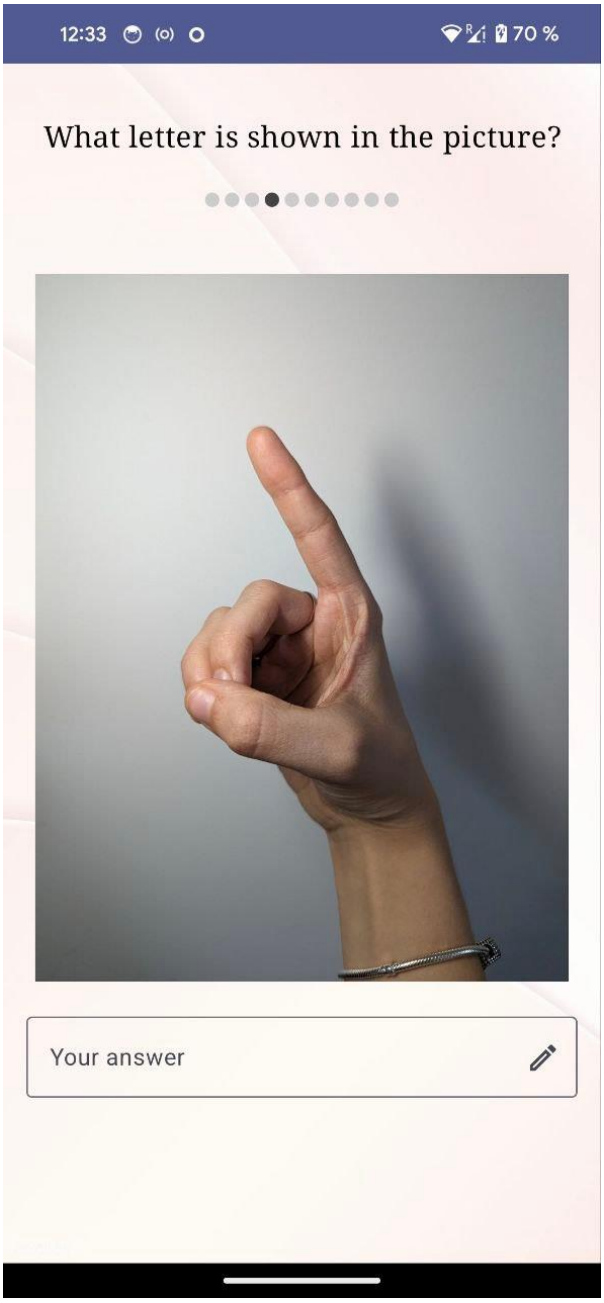

Рисунок 4.10 – Урок, де користувач вводить відповідь

Система виводить запитання, індикатор кількості запитань, зображення жесту та поле вводу відповіді. Якщо користувач ввів відповідь і перегорнув на наступне запитання, його відповідь зберігається. У кінці виводиться результат із усіма питаннями, але правильної відповіді система не видає, лише текст, чи правильно відповів користувач.

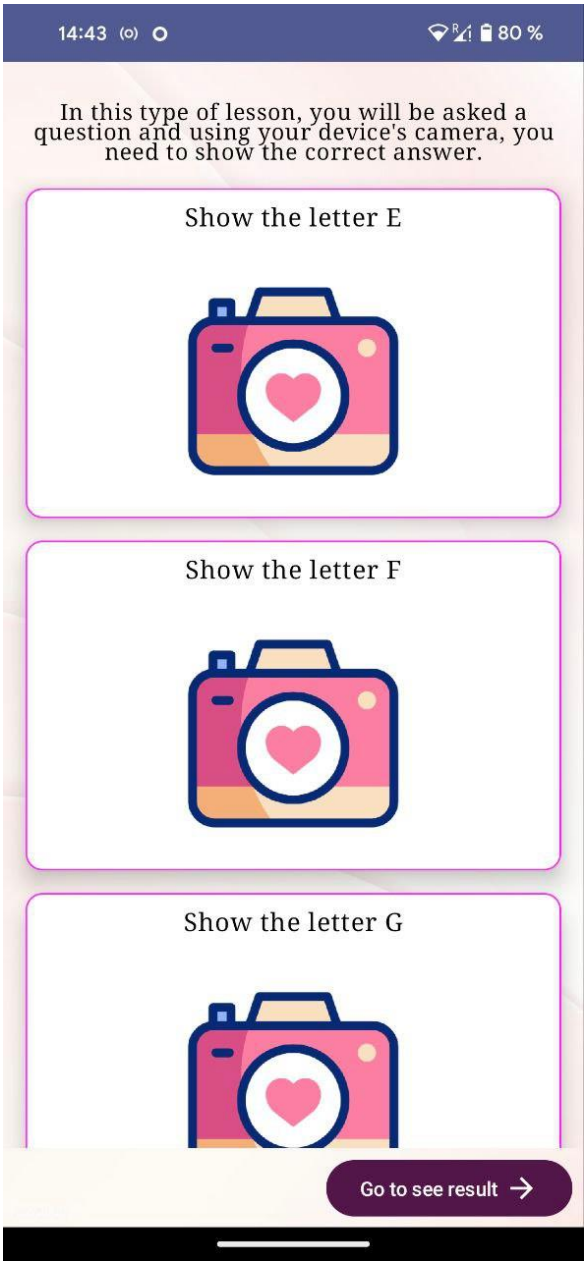

Рисунок 4.11 – Урок, де користувач показує відповідь

В цьому типі уроку, система задає питання, а користувач має сфотографувати свій жест. Після підтвердження користувача зображення у застосунку камери, воно передається до тесту для перевірки. Результат користувач побаче лише коли дасть відповіді на всі запитання. На рисунку нижче можна побачити заповнений тест із відповідями на запитання.

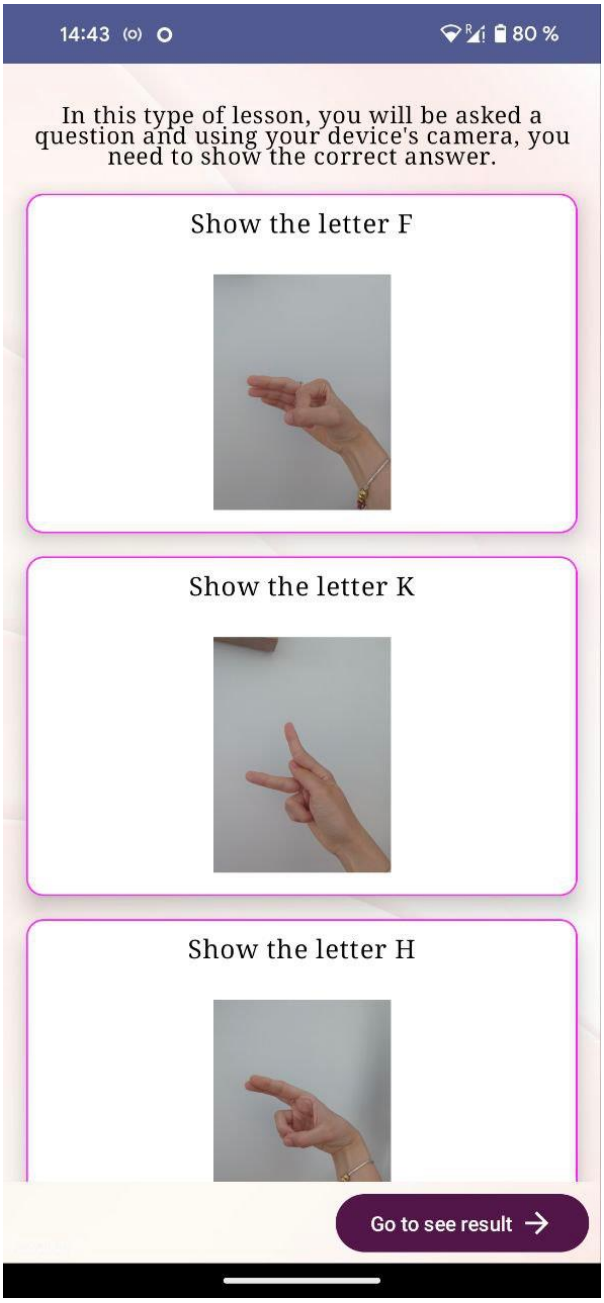

Рисунок 4.12 – Заповнений тест

Після натискання кнопки «Go to see result» система передає усі зображення до класу «ClassifyImage.kt» і, власне, штучний інтелект після цього аналізує зображення, генерує результати тесту і сама система виводить аналіз на екран (рис. 4.13).

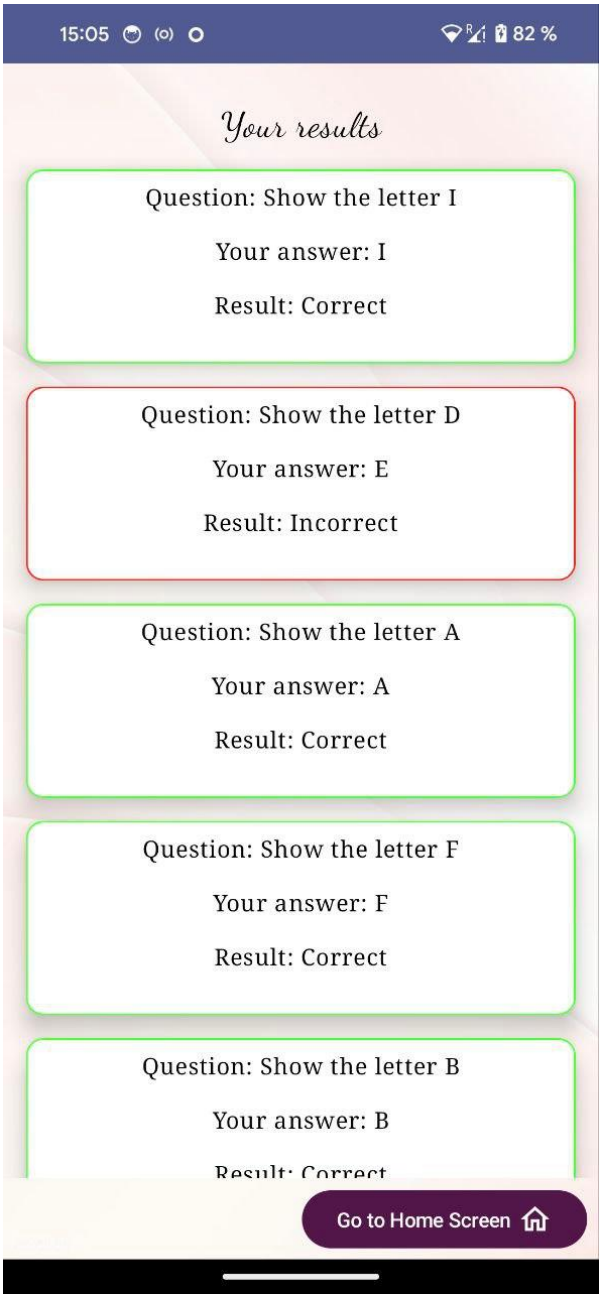

Рисунок 4.13 – Сторінка результату проходження тесту

Можна зробити висновок, що всі вище зазначені функції в застосунку взаємопов’язані і жити один без одного не можуть. Створено зручний та комфортний дизайн для того, щоб користувач, котрий використовував мобільний затосунок вивчення сурдоалфавіту, хотів повертатися до нього ще та ще.

4.2 Реалізація технології розпізнавання жестів

Одна із головних функцій в застосунку – це реалізація технології розпізнавання жестів. Використаємо для цього Google Teachable Machine, котра допомагає швидше та легко навчити модель. В нашому випадку це модель, котра містить літери англійського сурдоалфавіту. Для початку треба зібрати вибірку із фото руки для певних літер (A, B, C, D, E) у кількості від 10 до 20.

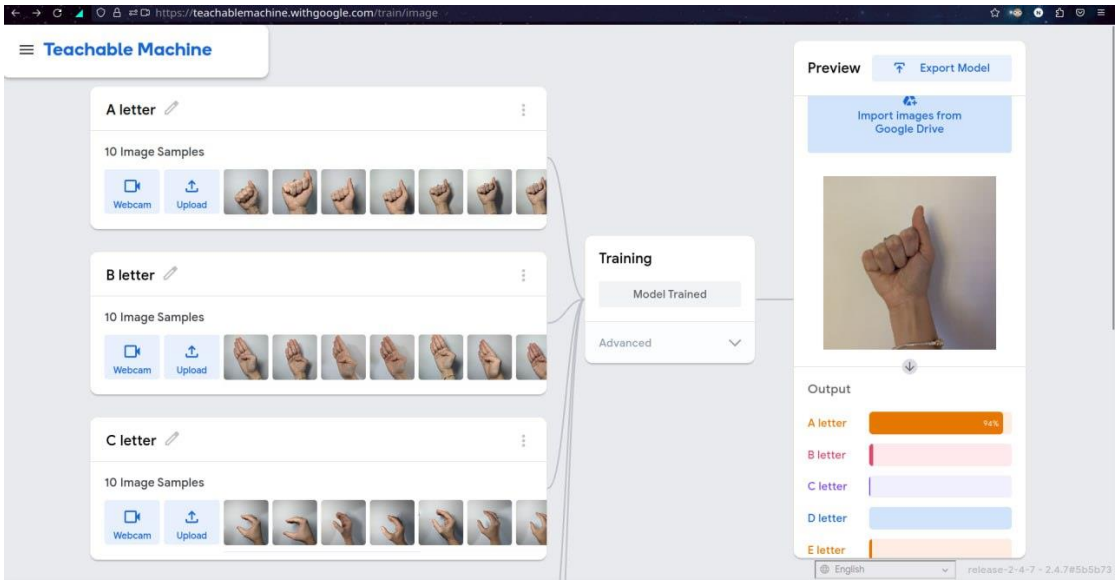

Рисунок 4.14 – Результат першого навчання моделі

Бачимо, що модель добре натренувалась із 5 літерами та видає більше ніж 90% схожості, наприклад із літерою A. Додаємо ще літер: F, G, H, I, K (рис. 4.15).

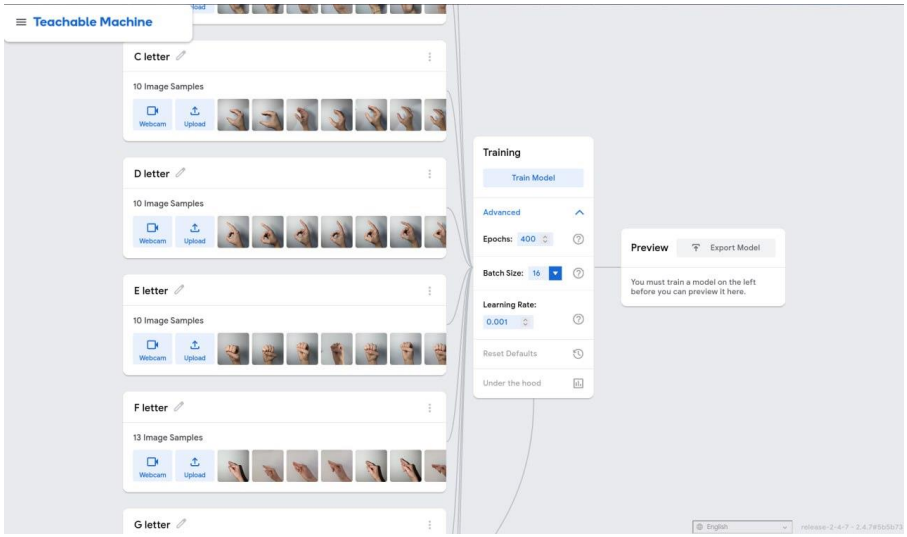

Рисунок 4.14 – Результат другого навчання моделі

Після цього впав відсоток схожості і такі літери як C, D, F, H не розпізнаються взагалі (мають малий відсоток схожості) або розпізнаються як інша літера. Експеримент проводився із 400 епохами. Епохи – це один цикл навчання на всьому датасеті. Тобто, модель проходилась по одній і тій самій виборці фото 400 разів. Потрібно розуміти, що кількість епох – це індивідуальний параметр, який залежить від розміру датасету та кількості нейронів в нейронній мережі. Якщо взяти замало епох, то відбудеться недонавчання (рис. 4.16) і мережа не навчиться розпізнавати класи правильно. Якщо взяти більше епох, то відбудеться перенавчання – коли мережа заучує тренувальні зображені в датасеті, а нові зображення вона буде класифікувати вже не правильно. Тому потрібно шукати золоту середину.

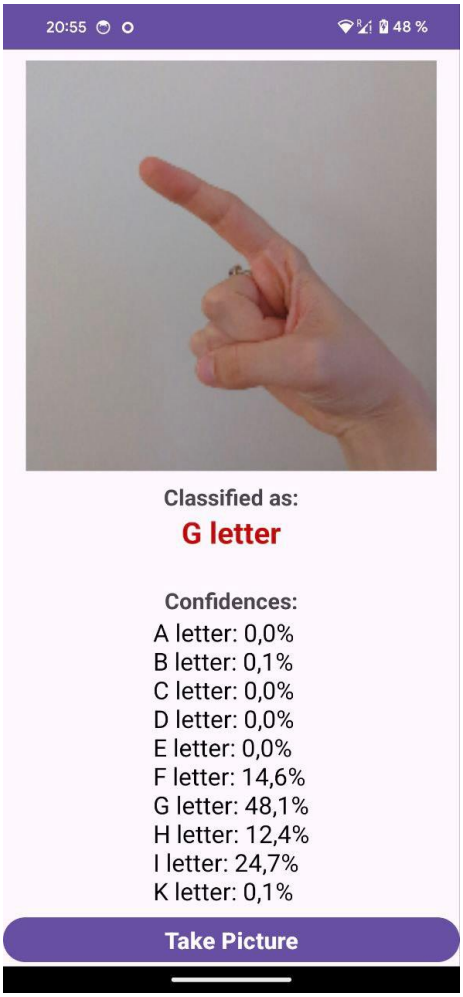

Рисунок 4.16 – Результат навчання моделі

На рисунку 4.16 добре видно, що відсоток розпізнавання дуже малий – 48.1%. Бажаний результат – більше ніж 90%. Є декілька кроків вирішення цієї проблеми:

1) до кожного класу літери додати фото без тіні та з нею;
2) збільшити кількість епох з 400 до 1000.

В результаті маємо успіхи: абсолютно усі літери розпізнаються із відсотком більше ніж 90. Для приклада візьмемо літеру В і результат тренування можна побачити на рисунку нижче.

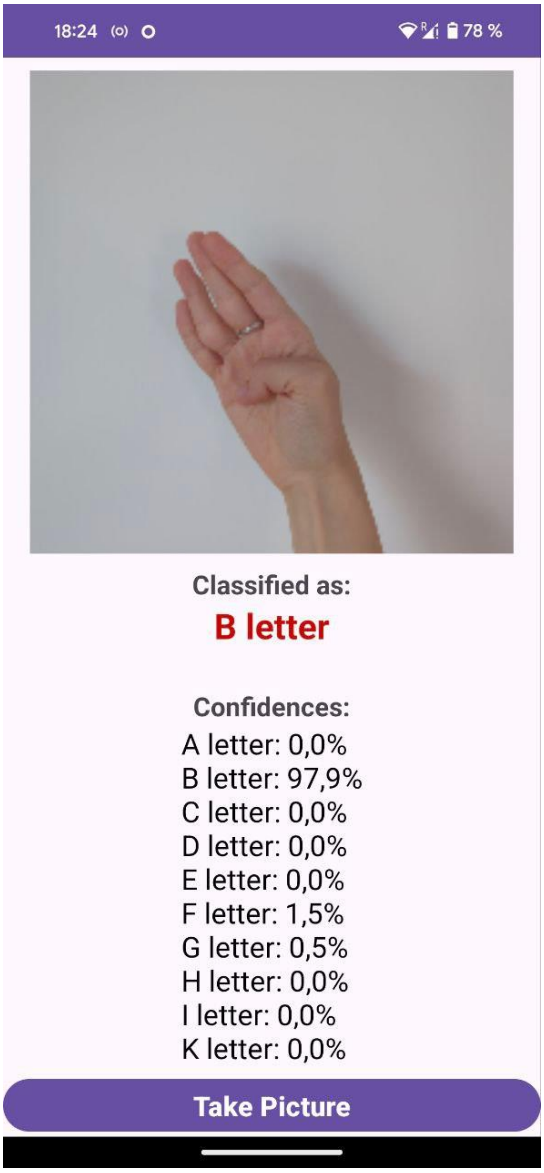

Рисунок 4.16 – Кінцевий результат

Отже, на даному етапі маємо готову, добренавчану модель. Залишився код на Kotlin під Android для того щоб мобільний застосунок вивчення сурдоалфавіту працював із цією технологією.

```
class ClassifyImage(
        private val context: Context
) {

    private val imageSize = 224
    private var result: String? = ""

    fun classifyImage(image: Bitmap) {
        try {
            val model = Model.newInstance(context)

            val inputFeature0 =
                    TensorBuffer.createFixedSize(intArrayOf(1, 224, 224,
3), DataType.FLOAT32)
            val byteBuffer = ByteBuffer.allocateDirect(4 * imageSize *
imageSize * 3)
            byteBuffer.order(ByteOrder.nativeOrder())

            val intValues = IntArray(imageSize * imageSize)
            image.getPixels(intValues, 0, image.width, 0, 0, im-
age.width, image.height)

            var pixel = 0
            for (i in 0 until imageSize) {
                for (j in 0 until imageSize) {
                    val `val` = intValues[pixel++] // RGB
                    byteBuffer.putFloat((`val` shr 16 and 0xFF) * (1f /
255f))
                    byteBuffer.putFloat((`val` shr 8 and 0xFF) * (1f /
255f))
                    byteBuffer.putFloat((`val` and 0xFF) * (1f / 255f))
                }
            }

            inputFeature0.loadBuffer(byteBuffer)

            val outputs = model.process(inputFeature0)
            val outputFeature0 = outputs.outputFeature0AsTensorBuffer

            val confidences = outputFeature0.floatArray
            var maxPos = 0
            var maxConfidence = 0f
            for (i in confidences.indices) {
                if (confidences[i] > maxConfidence) {
                    maxConfidence = confidences[i]
                    maxPos = i
                }
            }
            val classes = arrayOf(
```

```
                    "A letter",
                    "B letter",
                    "C letter",
                    "D letter",
                    "E letter",
                    "F letter",
                    "G letter",
                    "H letter",
                    "I letter",
                    "K letter"
            )
            Log.d("TESTV", classes[maxPos])

            var s: String? = ""
            for (i in classes.indices) {
                s += java.lang.String.format("%s: %.1f%%\n", classes[i],
confidences[i] * 100)
            }
            Log.d("TESTV - 2", s.toString())
            result = s

            model.close()
        } catch (exception: Exception) {
            Log.d("ClassifyImage", exception.message.toString())
        }
    }

}
```

Цей код використовується для підрахунку результату проходження тесту. І пізніше виводиться результат та аналіз на екрані «Results Screen».

4.3 Тестування застосунку

Перед випуском мобільного застосунку його слід ґрунтовно протестувати, щоб забезпечити його бездоганну роботу та приємний досвід користувачів. Завдяки тестуванню можна виявити та усунути всі недоліки та помилки до того, як користувачі почнуть користуватися застосунком.

Таблиця 4.1 – Тест 1. Завантаження застосунку

Назва тесту	Завантаження та запуск застосунку
Мета	Перевірити чи встановився та чи запускається застосунок
Пріоритет	High

Час виконання	2 хвилини
Дата	25.05.2024
Передумови	1) користувач натиснув на файл APK для встановлення; 2) смартфон відповідає мінімальним вимогам застосунку.
Дії	Натискання на іконку застосунку
Очікуваний результат	– застосунок успішно встановлений без помилок; – головний екран застосунку відображається.
Наявний результат	Успішний

Таблиця 4.2 – Тест 2. Реєстрація користувача

Назва тесту	Реєстрація користувача
Мета	Створити обліковий запис
Пріоритет	High
Час виконання	10 хвилин
Дата	25.05.2024
Передумови	Користувач не авторизований в системі.
Дії	Користувач відкриє застосунок.
Очікуваний результат	– користувач зайде в застосунок; – введе пошту та пароль (або обере логін за допомогою Google); – натисне кнопку «Login».
Наявний результат	Успішний

Таблиця 4.3 – Тест 3. Редагування облікового запису

Назва тесту	Редагування облікового запису
Мета	Налаштувати обліковий запис
Пріоритет	Medium

Час виконання	2 хвилини
Дата	25.05.2024
Передумови	Користувач авторизований в системі.
Дії	Натискання на кнопку «My Account».
Очікуваний результат	– користувач зайде в застосунок; – натискає кнопку «My Account»; – змінює пошту та/або фото профілю; – натискає кнопку «Update info».
Наявний результат	Успішний

Таблиця 4.4 – Тест 4. Здійснення пошуку курсу

Назва тесту	Пошук курсів
Мета	Знайти необхідний курс.
Пріоритет	Low
Час виконання	10 хвилини
Дата	25.05.2024
Передумови	Користувач авторизований в системі.
Дії	Натискання на кнопку «Search Courses».
Очікуваний результат	– користувач зайде в застосунок; – натискає кнопку «Search Courses»; – вводить пошуковий запит; – бачить перед собою необхідні курси.
Наявний результат	Успішний

Таблиця 4.5 – Тест 5. Вивчення уроку з використанням технології розпізнавання жестів

Назва тесту	Вивчення уроку з використанням технології розпізнавання жестів
Мета	Обрати урок із відповідною технологією.

Пріоритет	High
Час виконання	30 хвилини
Дата	25.05.2024
Передумови	Користувач авторизований в системі.
Дії	Натискання на кнопку «Search Courses».
Очікуваний результат	– користувач зайде в застосунок; – натискає кнопку «Search Courses»; – шукає необхідний курс та обирає його; – обирає урок із технологією розпізнавання жестів; – проходить урок: показує жест на відповідне питання; – бачить результат.
Наявний результат	Успішний

У ході проведення тестування було виявлено слабкі сторони застосунку та виправлено їх згідно з попитом користувача. Також, проведено тестування основних п’яти сценаріїв для перевірки коректності їх виконання.

Висновки до розділу 4

В четвертому розділі проведено кодування мобільного застосунку вивчення сурдоалфавіту, завдяки чому було закріплено навички застосування мови програмування Kotlin, базові бібліотеки Android (Android SDK), архітектури Clean Architecture, архітектурного патерну MVVM, бібліотеки Room, Retrofit, TensorFlow Lite. Розроблено адаптивний, зручний та легкий дизайн застосунку, котрий приваблює користувача до вивчення нового. Детально описано реалізацію технології розпізнавання жестів за допомогою Google Teachable Machine. Проведено тестування застосунку.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра спрощено процес вивчення сурдоалфавіту за рахунок розробник ПЗ – мобільного застосунку вивчення сурдоалфавіту з використанням розпізнавання жестів. Для досягнення поставленої мети необхідно виконати наступний перелік завдань:

1) дослідити предметну галузь та застосунки-аналоги;

2) розробити інтерфейс мобільного застосунку, який зрозумілим та зручним для користувачів будь-якого віку та можливостей;

3) проєктування та моделювання архітектури мобільного застосунку;

4) інтегрувати технологію розпізнавання жестів для можливості більш детального вивчення сурдоалфавіту;

5) забезпечити безпеку та конфіденційність даних користувачів, які використовують застосунок;

6) провести тестування та вдосконалення застосунку згідно з отриманими результатами.

У першому розділі бакалаврської кваліфікаційної роботи було проаналізовано існуючі аналоги Android-застосунку для вивчення сурдоалфавіту, вибрано три конкурентні застосунки та оцінено їхні переваги й недоліки. Також, проаналізовано розроблювану систему «EasyGestures», визначено типи користувачів, основні можливості та сценарії використання. Сформульовано вимоги до ПЗ, детально описано його призначення, функціональність, архітектуру та необхідні технології.

У другому розділі змодельовано та описано ПЗ за допомогою діаграм прецедентів, розгортання, послідовностей та діяльності. Це допоможе підтримувати ПЗ та додавати новий функціонал в Android-застосунок. Створено унікальний use case для реєстрації користувача, що виокремлює основні сценарії використання застосунка. Алгоритми роботи відображають взаємодію користувачів, включаючи пошук, вибір і вивчення курсів, забезпечуючи

зручність і ефективність. UML-діаграми чітко відображають структуру, вимоги та архітектуру ПЗ, сприяючи створенню стабільного та ефективного продукту, який відповідає потребам користувачів.

У третьому розділі виконано моделювання та конструювання Android-застосунку для вивчення сурдоалфавіту. Розроблено діаграму розгортання, яка демонструє архітектуру мобільного застосунку. Детально описано технології, архітектуру, архітектурний патерн та бібліотеки, які будуть використовуватися, та проведено їх аналіз. Представлено базову діаграму класів і детальну діаграму пакетів для кращого розуміння ієрархії архітектури застосунку. Закріплено навички створення UML-діаграм.

У четвертому розділі здійснено кодування мобільного застосунку для вивчення сурдоалфавіту, що дозволило закріпити навички програмування на Kotlin, використання базових бібліотек Android (Android SDK), архітектури Clean Architecture, архітектурного патерну MVVM, а також бібліотек Room, Retrofit і TensorFlow Lite. Розроблено адаптивний, зручний та легкий дизайн, який заохочує користувачів до вивчення сурдоалфавіту. Детально описано реалізацію технології розпізнавання жестів за допомогою Google Teachable Machine. Застосунок було ретельно протестовано для забезпечення його стабільності та функціональності.

Таким чином, створено ефективний інструмент для вивчення сурдоалфавіту, що поєднує сучасні технології та зручний інтерфейс, забезпечуючи позитивний досвід користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What is a Use Case. URL: https://www.wrike.com/blog/what-is-a-use-case/ (дата звернення: 03.05.2024);

2. Як і навіщо писати Use Cases. URL: https://dou.ua/lenta/articles/use-cases/ (дата звернення: 03.05.2024);

3. Splash Screens. URL: https://developer.android.com/develop/ui/views/launch/splash-screen (дата звернення: 05.02.2024);

4. Діаграма прецедентів. URL: https://uk.wikipedia.org/wiki/Діаграма_прецедентів (дата звернення: 03.05.2024);

5. Діаграма розгортання. URL: https://uk.wikipedia.org/wiki/Діаграма_розгортання (дата звернення: 03.05.2024);

6. Save simple data with SharedPreferences. URL: https://developer.android.com/training/data-storage/shared-preferences (дата звернення: 03.05.2024);

7. Read and Write Data on Android. URL: https://firebase.google.com/docs/database/android/read-and-write (дата звернення: 03.05.2024);

8. TensorFlow Lite for Android. URL: https://www.tensorflow.org/lite/android (дата звернення: 03.05.2024);

9. Діаграма послідовності. URL: https://uk.wikipedia.org/wiki/Діаграма_послідовності (дата звернення: 03.05.2024);

10. Діаграма діяльності. URL: https://uk.wikipedia.org/wiki/Діаграма_діяльності (дата звернення: 14.05.2024);

11. Kotlin. URL: https://uk.wikipedia.org/wiki/Kotlin (дата звернення: 14.05.2024);

12. Get started with Kotlin. URL: https://kotlinlang.org/docs/getting-started.html (дата звернення: 14.05.2024);

13. Implement Clean Architecture in Android. URL: https://medium.com/simform-engineering/clean-architecture-in-android-12d61c4f5318 (дата звернення: 14.05.2024);

14. MVVM (Model View ViewModel) Architecture Pattern in Android. URL: https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-android/ (дата звернення: 14.05.2024);

15. Google Firebase. URL: https://www.techtarget.com/searchmobilecomputing/definition/Google-Firebase (дата звернення: 14.05.2024);

16. Retrofit. A type-safe HTTP client for Android and Java. URL: https://square.github.io/retrofit/ (дата звернення: 14.05.2024);

17. Save data in a local database using Room. URL: https://developer.android.com/training/data-storage/room (дата звернення: 14.05.2024);

18. TensorFlow Lite for Android. URL: https://www.tensorflow.org/lite/android (дата звернення: 14.05.2024);

19. Діаграма класів. URL: https://uk.wikipedia.org/wiki/Діаграма_класів (дата звернення: 14.05.2024);

20. Database model. URL: https://en.wikipedia.org/wiki/Database_model (дата звернення: 14.05.2024)

21. Діаграма пакетів. URL: https://uk.wikipedia.org/wiki/Діаграма_пакетів (дата звернення: 14.05.2024);

22. Teachable Machine. URL: https://teachablemachine.withgoogle.com/ (дата звернення: 15.05.2024);

23. Kotlin Coroutines on Android. URL: https://developer.android.com/kotlin/coroutines (дата звернення: 15.05.2024).