SecureSphere

Mubeen Yaqub (BCS203165)

Damil Shahzad (BCS203068)

Faizan Ali (BCS203106)

Spring-2024

Ms. Tayyaba Zaheer

Department of Computer Science

Capital University of Science & Technology, Islamabad

Submission Form for Final-Year

PROJECT REPORT

Version	V 1.0

NUMBER OF MEMBERS	3

TITLE	SecureSphere

SUPERVISOR NAME	Ms. Tayyaba Zaheer

MEMBER NAME	REG. NO.	EMAIL ADDRESS
Mubeen Yaqub	BCS203165	bcs203165@cust.pk
Faizan Ali	BCS203106	bcs203106@cust.pk
Damil Shahzad	BCS203068	bcs203068@cust.pk

MEMBERS' SIGNATURES		Supervisor's Signature
Mubeen Yaqub (BCS203165)_____ Faizan Ali(BCS203106)_____ Damil Shahzad (BCS203068)_____		

APPROVAL CERTIFICATE

This project, entitled as “Secure Sphere” has been approved for the award of

Bachelor of Science in Computer Science

Committee Signatures:

Supervisor: _____

(Ms. Tayyaba Zaheer)

Project Coordinator: _____

(Mr. Bilal Ahmed)

Head of Department: _____

(Dr. Abdul Basit)

DECLARATION

We, hereby, declare that “No portion of the work referred to, in this project has been submitted in support of an application for another degree or qualification of this or any other university/institute or other institution of learning”. It is further declared that this undergraduate project, neither as a whole nor as a part thereof has been copied out from any sources, wherever references have been provided.

MEMBERS’ SIGNATURES

(Mubeen Yaqub)

(Damil Shahzad)

(Faizan Ali)

ACKNOWLEDGEMENTS

We cordially thank our supervisor Ms. Tayyaba Zaheer for her valuable time and efforts throughout this project. We thank our panel teachers who helped us in identifying the mistakes and helped us through this tough time. We heartedly thank our family who support and motivate us during this phase. Without help of above-mentioned people this project was not possible.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1. Project Introduction

Many people have CCTV cameras installed in their homes, from where they can check the live feed of their homes. But there are no alerts, in case someone is observing your house, or an intruder has entered your premises. Normally these types of security systems are designed for organizations or governments, but not for domestic purposes. That's why we have decided to implement this idea, so that a normal person can have access to these security features at the tip of their fingers, at an affordable price. The CCTV feed will be monitored with our machine-learning algorithms, in case a suspicious person is detected outside our house (strolling in the premises for 3 minutes let's say), then an emergency alert will be generated on the users' mobile application. In case an intruder bypasses our boundaries, then an emergency alert will be generated so that the user can take action before too much damage is done. Moreover, our application will be linked with a database, that will store all the user data and facial recognition data of familiar and unfamiliar faces (the user will be given an option in the app to select the familiar faces and add them to favorites). We also plan to keep a log of times, i.e. how many times a person has visited our home for unidentified persons.

1.2. Existing Examples / Solutions

Table 1. Existing Solutions Comparison

Features	SecureSphere	Ring	Nest (Google)	Arlo	SimpliSafe	ADT
Live CCTV feed access	✔	✔	✔	✔	✔	✔
Mobile app for remote monitoring/alerts	✔	✔	✔	✔	✔	✔
Motion detection/alerts	✔	✔	✔	✔	✔	✔
Real-time monitoring using machine learning	✔	✔	✘	✘	✘	✘
Trespassing alerts	✔	✘	✘	✘	✘	✘
Facial recognition	✔	✘	✘	✘	✘	✘
Log of visitor activity	✔	✘	✘	✘	✘	✘
Database storage of user data	✔	✔	✔	✔	✔	✔
Integration of familiar/unfamiliar faces in database	✔	✘	✘	✘	✘	✘
Customizable alerts for emergency situations	✔	✔	✔	✔	✔	✔

1.3. Business Scope

While the primary focus of our project is to enhance home security, its potential extends to warehouses, offices, schools, and various organizations. Consider this scenario: in a household with only a few residents, ensuring their safety around the clock is paramount.

Turning our attention to organizations, many of them operate on fixed schedules, closing up shop at the end of the workday. What happens if someone attempts a break-in during off-hours? For many businesses with valuable assets and sensitive documents, the risk of theft is simply too great. To address this concern, we offer a valuable feature: the ability to set specific time periods during which your security cameras remain vigilant and send alerts. This way, you can avoid unnecessary alarms during the busy times when clients are coming and going.

This entire system hinges on the capabilities of the cameras you already have in place. However, if your existing cameras lack the necessary API integration, we also offer affordable hardware solutions that seamlessly integrate with your current setup. This allows you to continue using your cameras' live feed features while simultaneously gaining access to our security app.

1.4. Useful Tools and Technologies

1) Technologies

a) Python (most used language for ML and also easy to implement)

i) Machine Learning Libraries:

(1) TensorFlow (deep learning framework)

(2) Scikit-learn (ML framework)

(3) OpenCV (Computer Vision)

(4) MediaPipe

b) Flutter (cross platform development and great online community)

ii) Dart:

(1) User Interface

(2) Real-time alerts

(3) Integration with ML model

c) Back-end

i) Flask

d) Database (most reliable and widely used)

i) Firebase

2) Tools

a) IDEs

i) Visual Studio Code (Open-source light weight editor)

ii) PyCharm (best for python programming)

iii) Android Studio (Open-source)

iv) Google Colab (free and provides pre-installed libraries)

b) Version Control

i) Git

ii) GitHub (free to use)

c) Communication

i) Slack (free to use)

ii) Trello (free and great for assigning tasks)

iii) OneDrive (for storing key documents)

3) Datasets

a) Facial Detection

i) Human Detection Dataset (CCTV footage of humans)

ii) Labeled Faces in the Wild Home (5,749 folders of different people)

1.5. Project Work Break Down

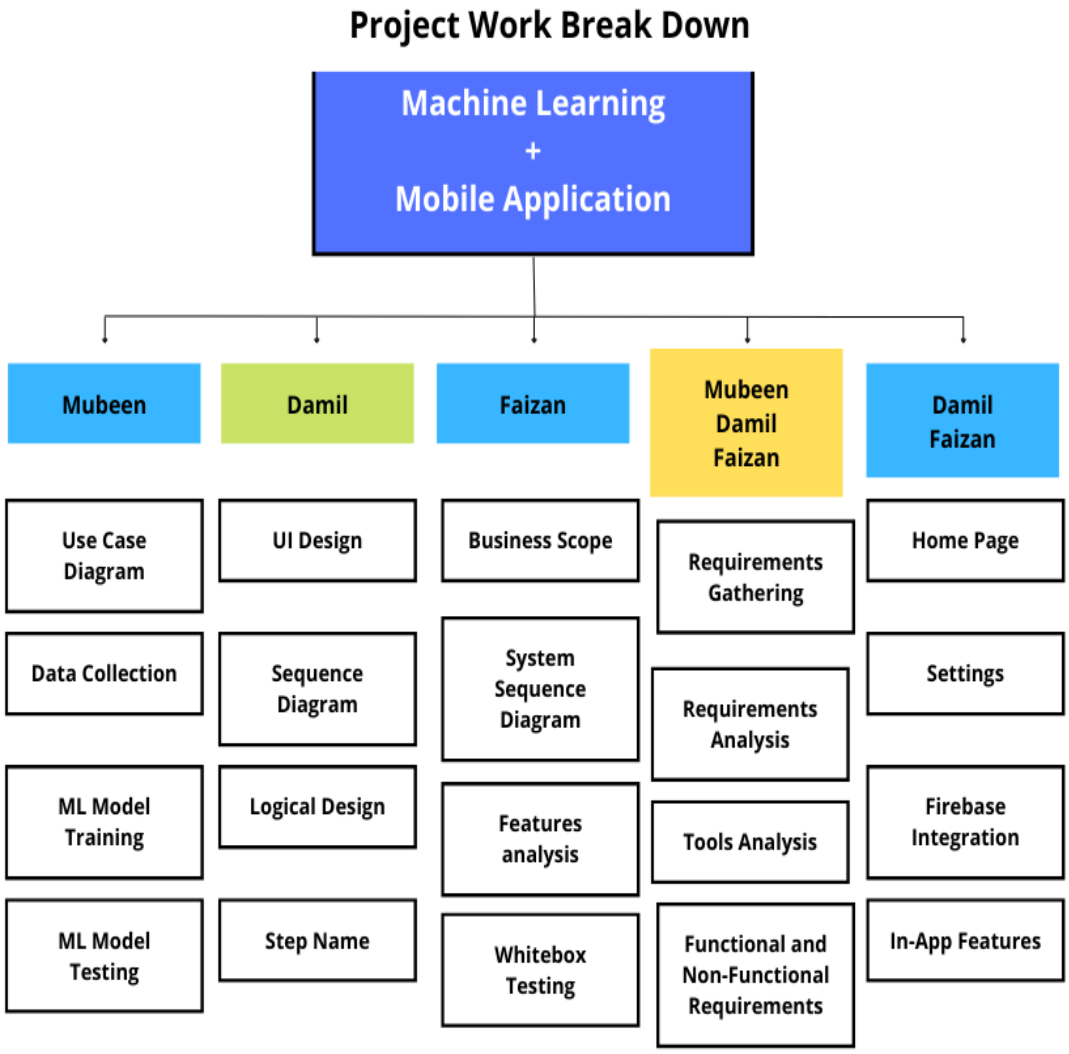


Figure 1. Project Work Break Down

1.6. Project Timeline

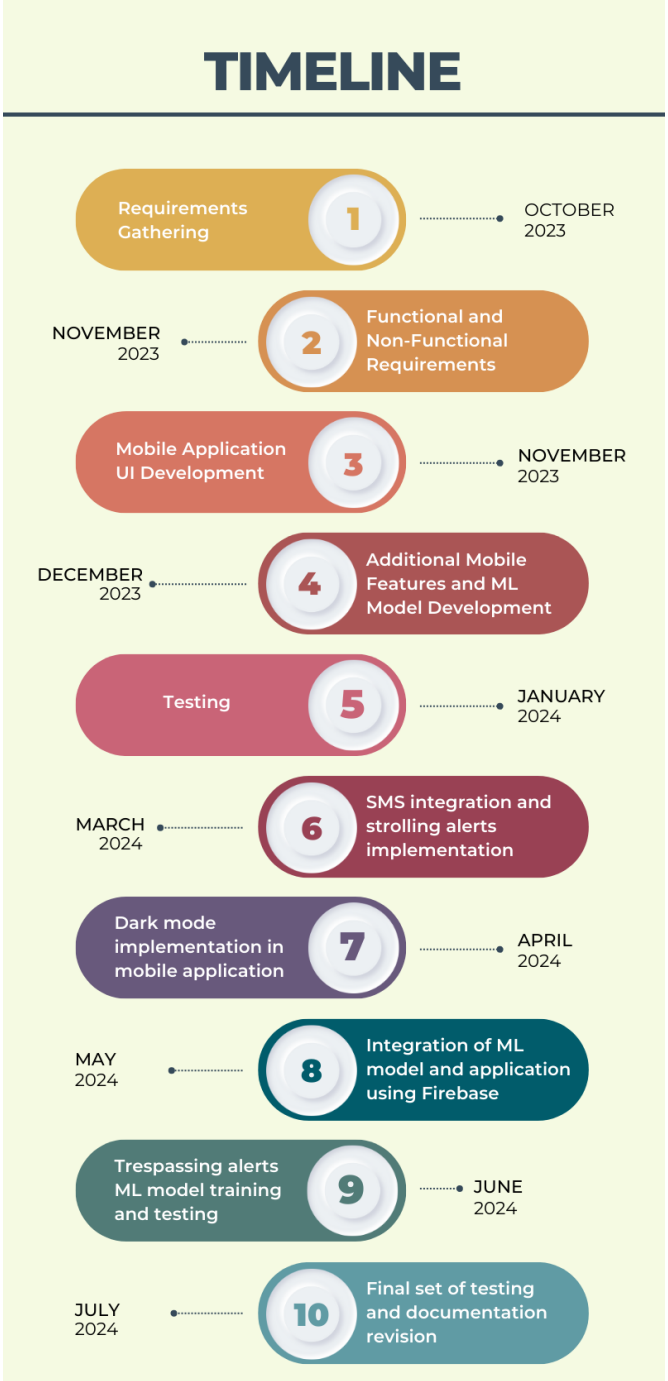


Figure 2. Project Timeline

1.7. Project Flow Diagram

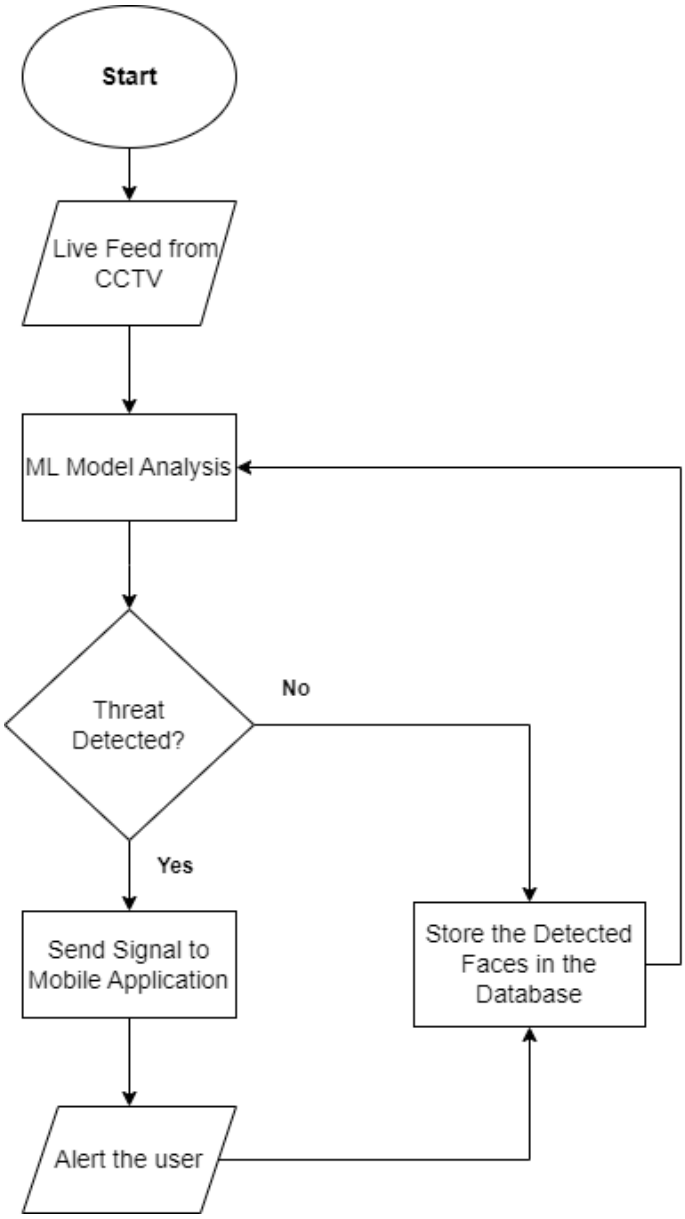


Figure 3. Project Flow Diagram

Chapter 2

Requirement Specifications and Analysis

Requirement Specification

Functional requirements are documented to list all set of operations and functionalities which software must perform. Whereas non-functional requirements describe some constraints on functional requirements to ensure the quality of the software. Use case diagram shows possible interactions between actor and system to get some result from the system. System sequence diagram represents the events that occur while interaction of actor with system in a specific use case.

2.1. Functional Requirements

In this table each functional requirement is assigned a serial number which will be used as reference. Status gives the information weather the requirement is completed, pending or is currently selected for development.

Table 2. Functional Requirements

Machine Learning	Serial Number	Functional Requirements	Type	Status
	1	System will provide suspicious movement detection	Core	Implemented
	2	Trespassing alerts	Core	Implemented
	3	Facial detection	Core	Implemented
	4	Suspicious strolling alert	Core	Implemented

Mobile Application	Serial Number	Functional Requirements	Type	Status
	1	The user can sign up	Core	Implemented
	2	The user can log in	Core	Implemented
	3	The user can see the facial logs	Core	Implemented
	4	The user can update the logs	Core	Implemented
	5	The use can toggle between light and dark mode	Core	Implemented
	6	The user will receive emergency alerts on threat detection	Core	Implemented
	7	The user can see live camera feed	Core	Implemented
	8	The user can name the saved faces	Core	Implemented
	9	The user can track how many times someone was detected	Core	Implemented
	10	User can log out	Core	Implemented
	11	The user can login using Gmail or Email	Core	Implemented
	12	The user will receive SMS alerts	Core	Implemented

2.2. Non-Functional Requirements

Non-functional requirements may specify some constraints that how a specific or set of functionalities should be achieved for the software.

Table 3. Non-Functional Requirements

Machine Learning	**Serial Number**	**Non-Functional Requirements**	**Category**
	1	Model Training and testing	Performance Efficiency
	2	Model Accuracy	Performance Efficiency
	3	Latency checkups	Performance Efficiency

Machine Learning	**Serial Number**	**Non-Functional Requirements**	**Category**
	1	Login should take <= 3 seconds	Performance Efficiency
	2	Password should be of at least 6 characters	Security
	3	Application will need active internet connection	Availability
	4	The user will have to verify their email	Data Integrity
	5	The password will have complexity requirements	Security
	6	User friendly interface	Performance Efficiency
	7	Compatibility with various operating systems	Availability Usability
	8	Maintain cost-effectiveness	Affordability
	9	Instant alerts on threat detection	Response Time

2.3. System Use Case Diagram

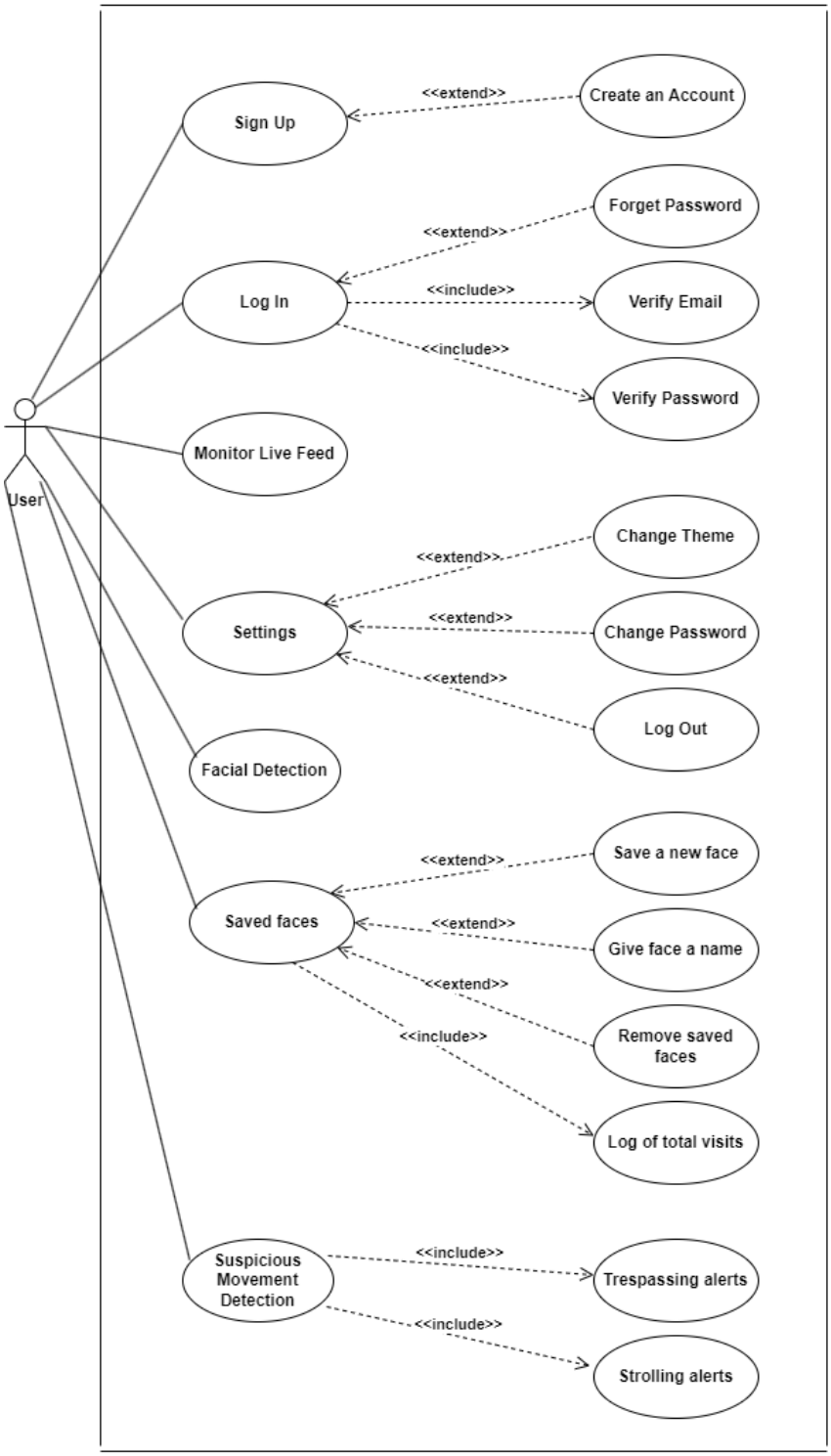


Figure 4. System Use Case Diagram

Use Case 1: Sign up

Use Case ID:	UC1		
Use Case Name:	Sign up		
Created By:	Mubeen Yaqub	**Last Updated By:**	Mubeen Yaqub
Date Created:	27/11/2023	**Last Revision Date:**	27/11/2023
Actors:	User		
Description:	User can sign up using their email, password or Facebook/Google accounts.		
Trigger:	Sign up button		
Preconditions:	User provides the email, password, and clicks the sign up button.		
Post conditions:	User will be signed up and be able to use the system.		
Normal Flow:	**User**	**System**	
	1. User clicks the sign up button. 3. User fills their credentials to create the account.	2. System provides the sign up form. 4. System signs up the user and saves their data in database.	
Alternative Flows:	User cancels the current form.		
Exceptions:	1. The database is not responding. 2. Form is incomplete. 3. User is not connected to the internet.		

Use Case 2: Log in

Use Case ID:	UC2		
Use Case Name:	Log in		
Created By:	Mubeen Yaqub	**Last Updated By:**	Mubeen Yaqub
Date Created:	27/11/2023	**Last Revision Date:**	27/11/2023
Actors:	User		
Description:	User can log in the app by using their valid credentials.		
Trigger:	Log in button		
Preconditions:	The user enters their credentials and clicks on log in button.		
Post conditions:	User gets logged in and can use the system.		
Normal Flow:	**User**	**System**	
	1. User clicks the log in button. 3. User fills their credentials to log in to the account.	2. System provides the log in form. 4. System logs in the user.	
Alternative Flows:	User cancels the current form.		
Exceptions:	1. The database is not responding. 2. Form is incomplete. 3. User is not connected to the internet.		

Use Case 3: Monitor Live Feed

Use Case ID:	UC3		
Use Case Name:	Monitor Live Feed		
Created By:	Mubeen Yaqub	**Last Updated By:**	Mubeen Yaqub
Date Created:	27/11/2023	**Last Revision Date:**	27/11/2023
Actors:	User		
Description:	The user can see the live camera feed with in the app.		
Trigger:	Home Page button		
Preconditions:	User is logged in.		
Post conditions:	User can watch the live feed.		
Normal Flow:	**User**	**System**	
	1. User clicks the home page button.	2. System fetches the live camera feed.	
Alternative Flows:	None		
Exceptions:	1. The database is not responding. 2. Cameras are not working. 3. User is not connected to the internet.		

Use Case 4: Settings

Use Case ID:	UC4		
Use Case Name:	Settings		
Created By:	Mubeen Yaqub	**Last Updated By:**	Mubeen Yaqub
Date Created:	27/11/2023	**Last Revision Date:**	27/11/2023
Actors:	User		
Description:	The user can change the theme, log out, and change password.		
Trigger:	Settings button		
Preconditions:	**User is logged in.**		
Post conditions:	User can change the relevant settings.		
Normal Flow:	**User**	**System**	
	1. User clicks on the settings button. 3. User makes the changes.	2. User is provided with the available options. 4. System either makes changes to the app or to the database.	
Alternative Flows:	None		
Exceptions:	1. The database is not responding. 2. Software bug. 3. User is not connected to the internet.		

Use Case 5: Facial Detection

Use Case ID:	UC5		
Use Case Name:	Facial Detection		
Created By:	Mubeen Yaqub	**Last Updated By:**	Mubeen Yaqub
Date Created:	27/11/2023	**Last Revision Date:**	27/11/2023
Actors:	Camera		
Description:	Person face will be detected and recorded in the database.		
Trigger:	Appearance of a face in the camera frame.		
Preconditions:	Face detected by a camera.		
Post conditions:	Face is detected and saved in the database.		
Normal Flow:	**Camera**	**System**	
	1. Camera detects a face	2. System decides if it's a face and stores it in the database.	
Alternative Flows:	None		
Exceptions:	1. The database is not responding. 2. Cameras are not working. 3. User is not connected to the internet.		

Use Case 6: Saved Faces

Use Case ID:	UC6		
Use Case Name:	Saved Faces		
Created By:	Mubeen Yaqub	**Last Updated By:**	Mubeen Yaqub
Date Created:	27/11/2023	**Last Revision Date:**	27/11/2023
Actors:	User		
Description:	A log of saved faces will be maintained inside the app.		
Trigger:	Face Log button		
Preconditions:	User is logged in.		
Post conditions:	User can see the faces, delete faces, and update names of faces.		
Normal Flow:	**User**	**System**	
	1. User clicks on face log button. 3. User takes necessary actions.	2. System fetch the saved faces from the database. 4. System updates the database as per the user actions.	
Alternative Flows:	None		
Exceptions:	1. The database is not responding. 2. Software bug. 3. User is not connected to the internet.		

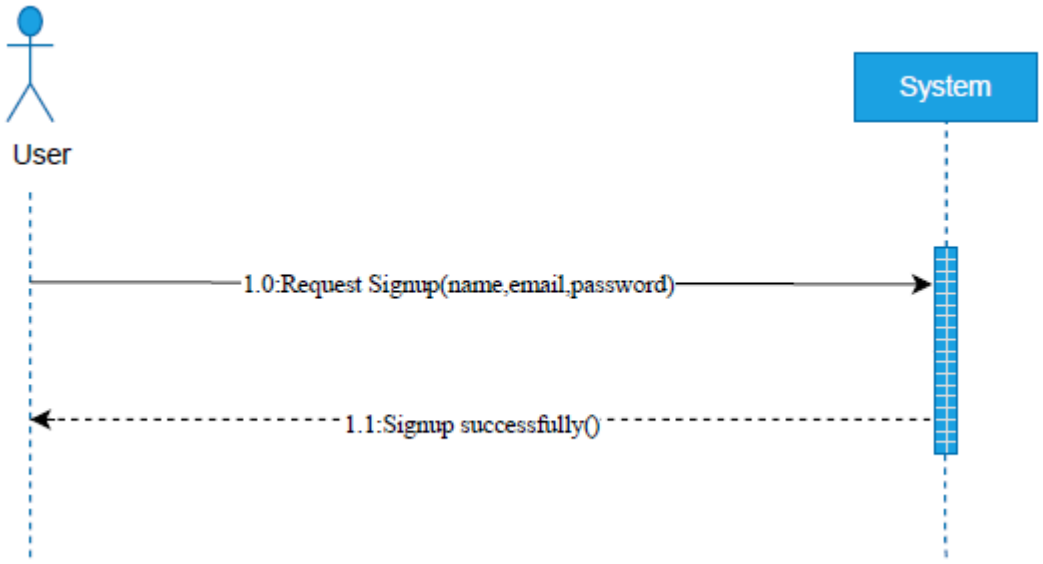
Use Case 7: Suspicious Movement Detection

Use Case ID:	UC7		
Use Case Name:	Suspicious Movement Detection		
Created By:	Mubeen Yaqub	Last Updated By:	Mubeen Yaqub
Date Created:	27/11/2023	Last Revision Date:	27/11/2023
Actors:	Camera		
Description:	The user will get alerts if an unknown face is detected and is present inside the camera frame for more than 2 minutes.		
Trigger:	Unknown face detected for more than 2 minutes.		
Preconditions:	Unknown face detected and remains in the frame for more than 2 minutes.		
Post conditions:	Alert is generated.		
Normal Flow:	**Camera**	**System**	
	1. Camera detects a face.	2. System calculates the duration of detection and generates an alert.	
Alternative Flows:	None		
Exceptions:	1. The database is not responding. 2. Software bug. 3. User is not connected to the internet.		

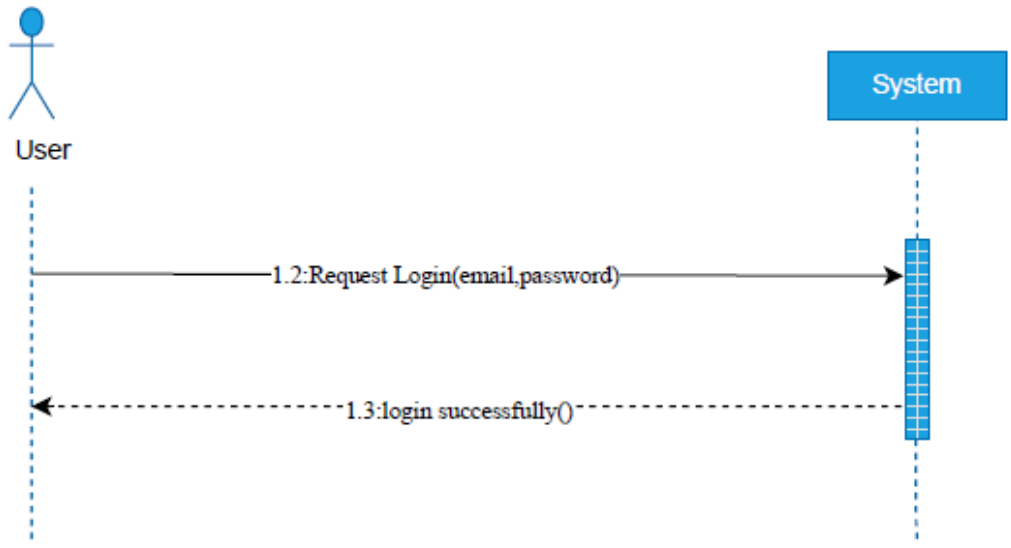
2.4. System Sequence diagram

System Sequence diagrams show the user and system interaction. System Sequence Diagram is a Sequence Diagram that shows, for a particular scenario of a use case, the events that actors generate, and possible inter-system events

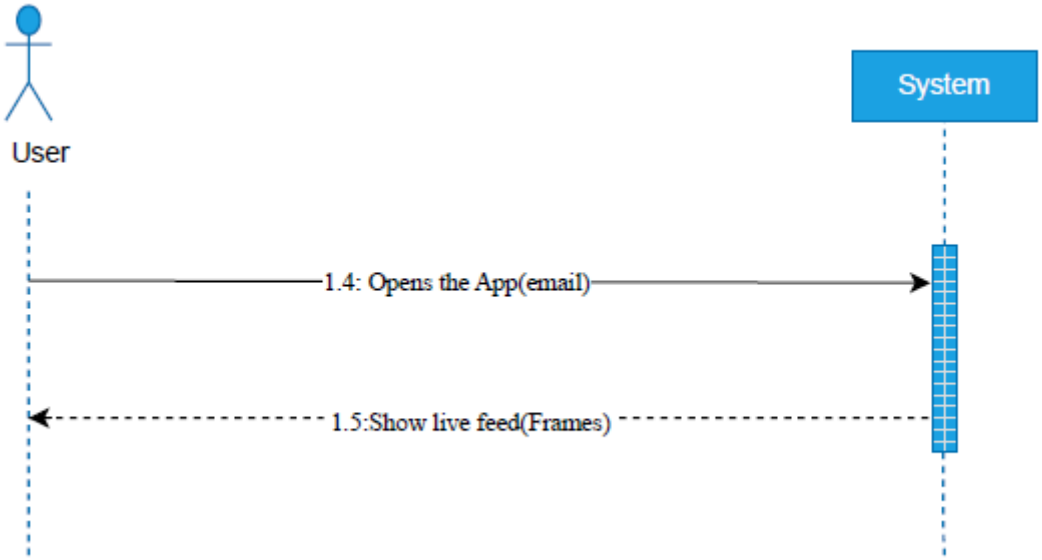
- **User Sign up**



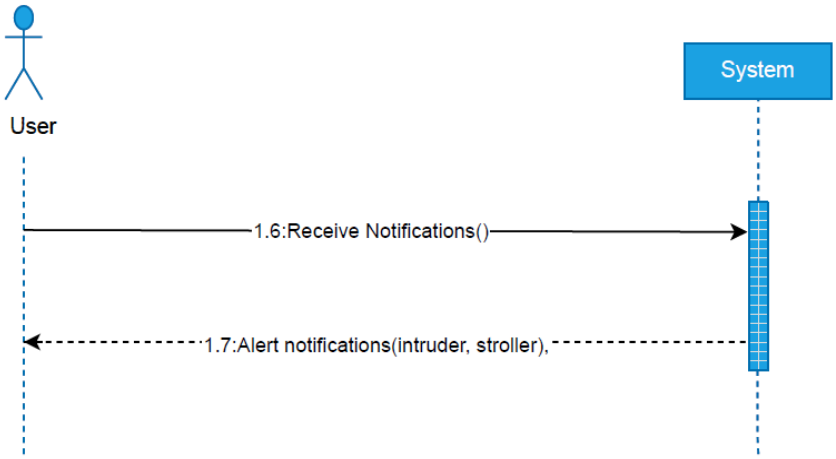
- **User log in**



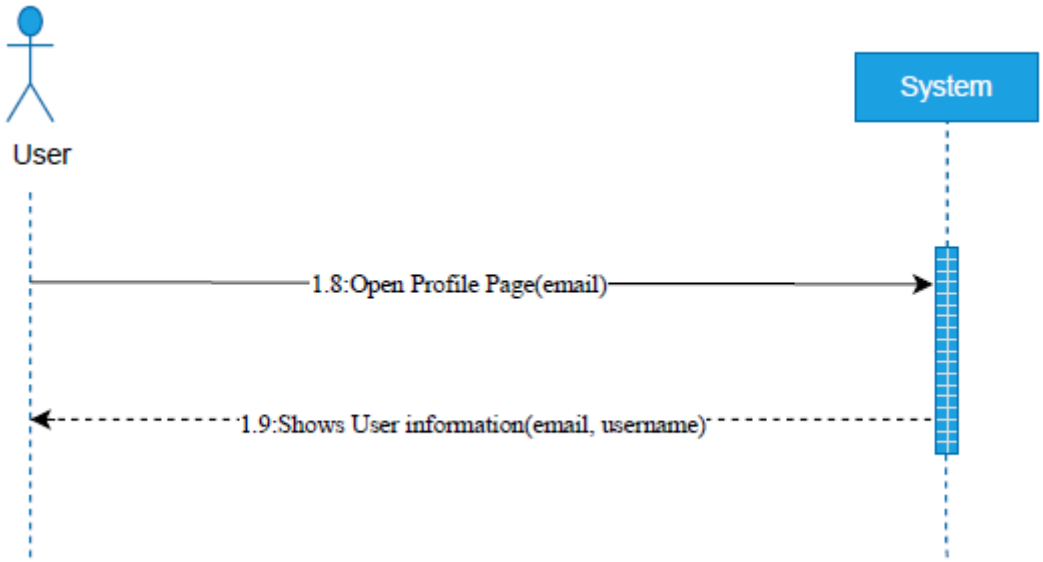
- **Shows live feed in homepage**



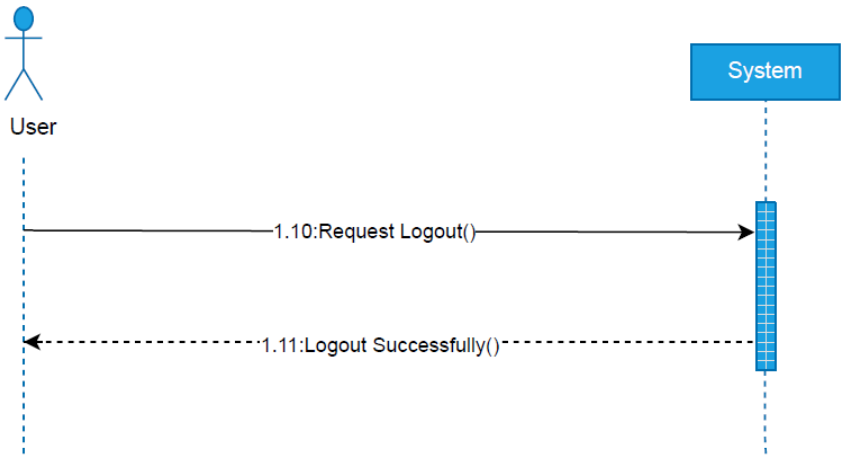
- **Notification page**



- **Profile Page**



- **Log out**



2.5. Domain Model

The basic concepts of the domain are user, camera, facial detection, database and alert are shown in the following figure representing domain model.

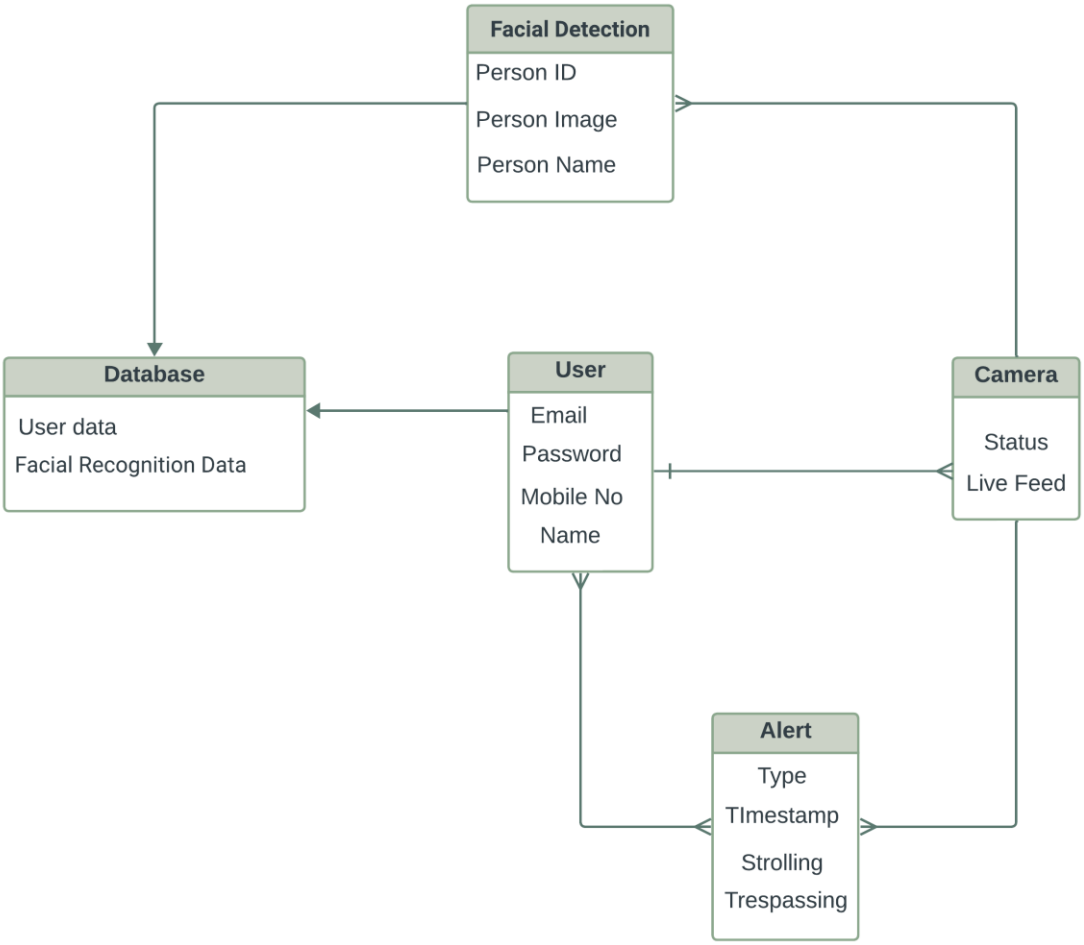


Figure 5. Domain Model

Chapter 3

System Design

This chapter is here to help the project along. Having a good design is crucial because without it, the project won't work well or meet quality standards. But it's not just about creating the design; it's also important to explain it to the people involved in the project. Communicating the architecture to everyone involved is just as important as coming up with the design in the first place. So, this chapter emphasizes that both creating a good design and making sure everyone understands it are key to making the project successful.

3.1. Layer Definition

Table 4. Design Layer Definition

Layer	Description
Presentation Layer	This layer is designed for engaging with users through a graphical user interface, facilitating interactive experiences.
Business Logic Layer	Within this layer lies the core business logic, encompassing both constraints and the majority of functions essential to the system's operation.
Database Layer	In this layer, you'll find the application's database, where all the relevant data is stored and managed.

3.1.1. Presentation Layer

Situated at the highest level, this layer exhibits information pertaining to the services accessible on a website. It connects with other layers by transmitting outcomes to the browser and other tiers within the network.

3.1.2. Business Logic Layer

Referred to as the Application Layer, middle tier, logic tier, or business logic, this stratum is separated from the presentation tier. It manages application functionality by executing detailed processing.

3.1.3. Database Layer

Comprising database servers for storing and retrieving information, this layer ensures data independence from application servers or business logic.

3.2. Software Architecture

Software architecture is described as the organization or structure of a system, where the system represents a collection of components that accomplish a specific function or set of functions. Below is the architecture diagram of the system:

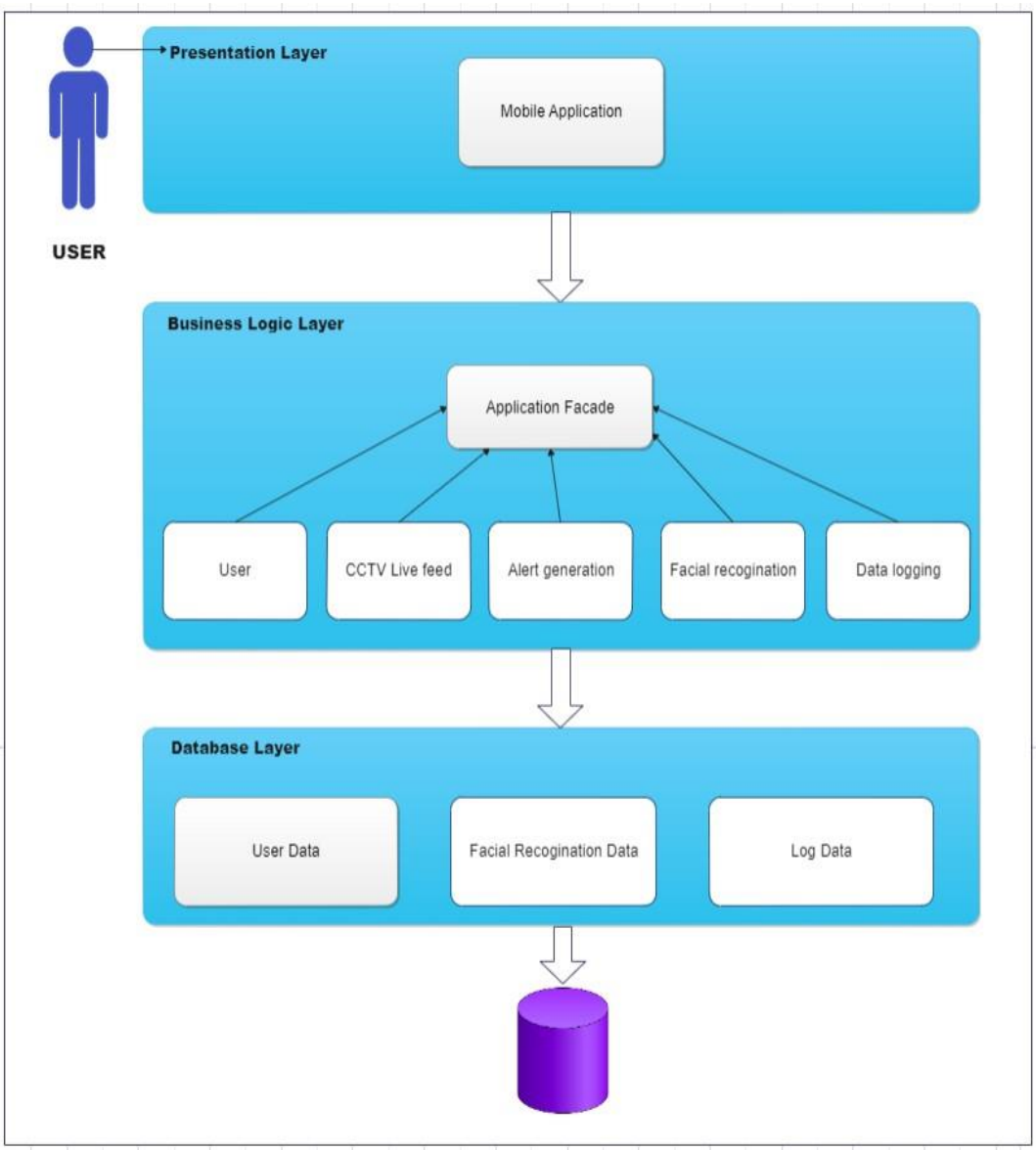


Figure 6. Software Architecture

3.3. Class Diagram

The class diagram describes the attributes and operations of a class and the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. The class diagram shown below is only for the selected requirements.

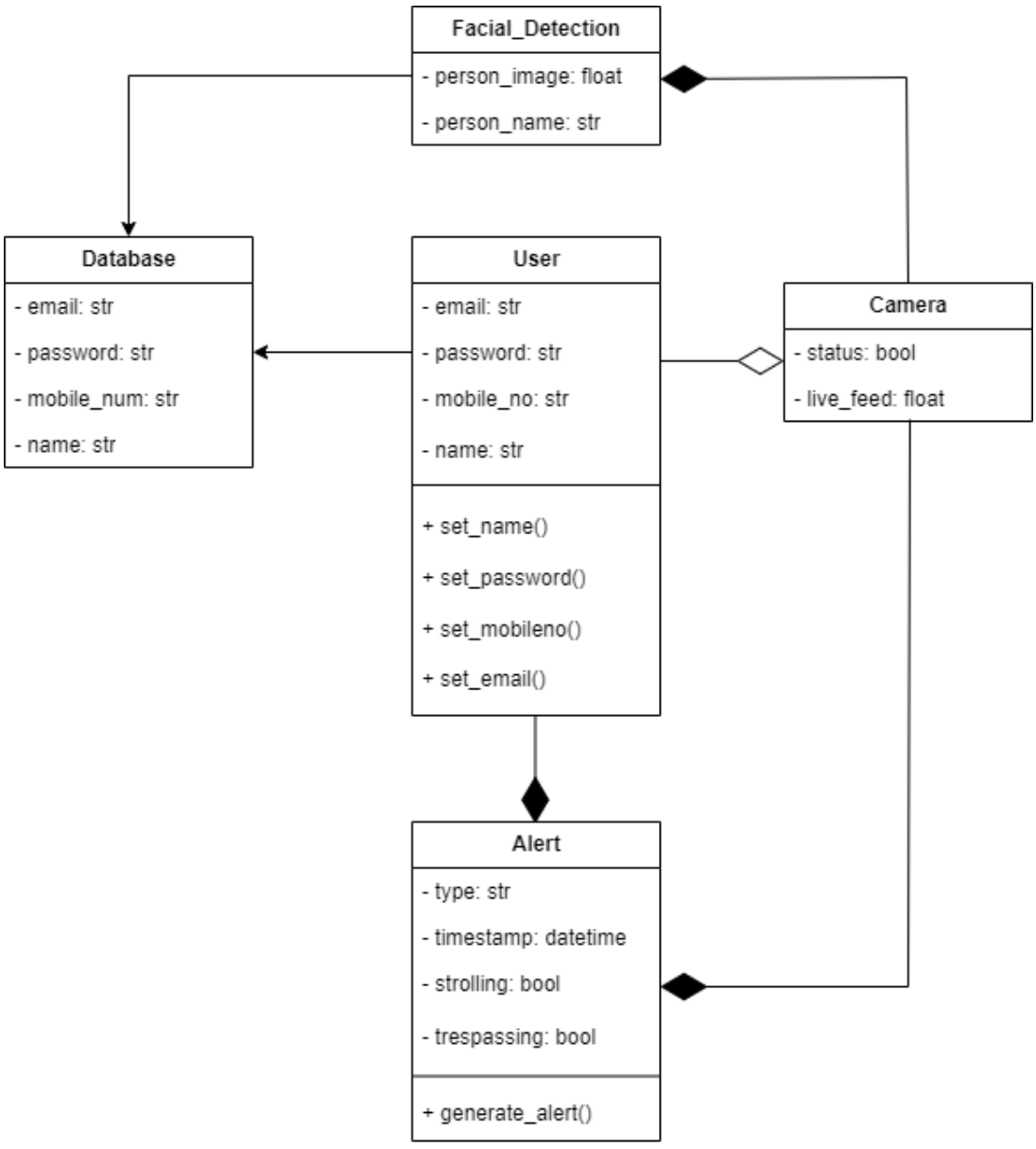
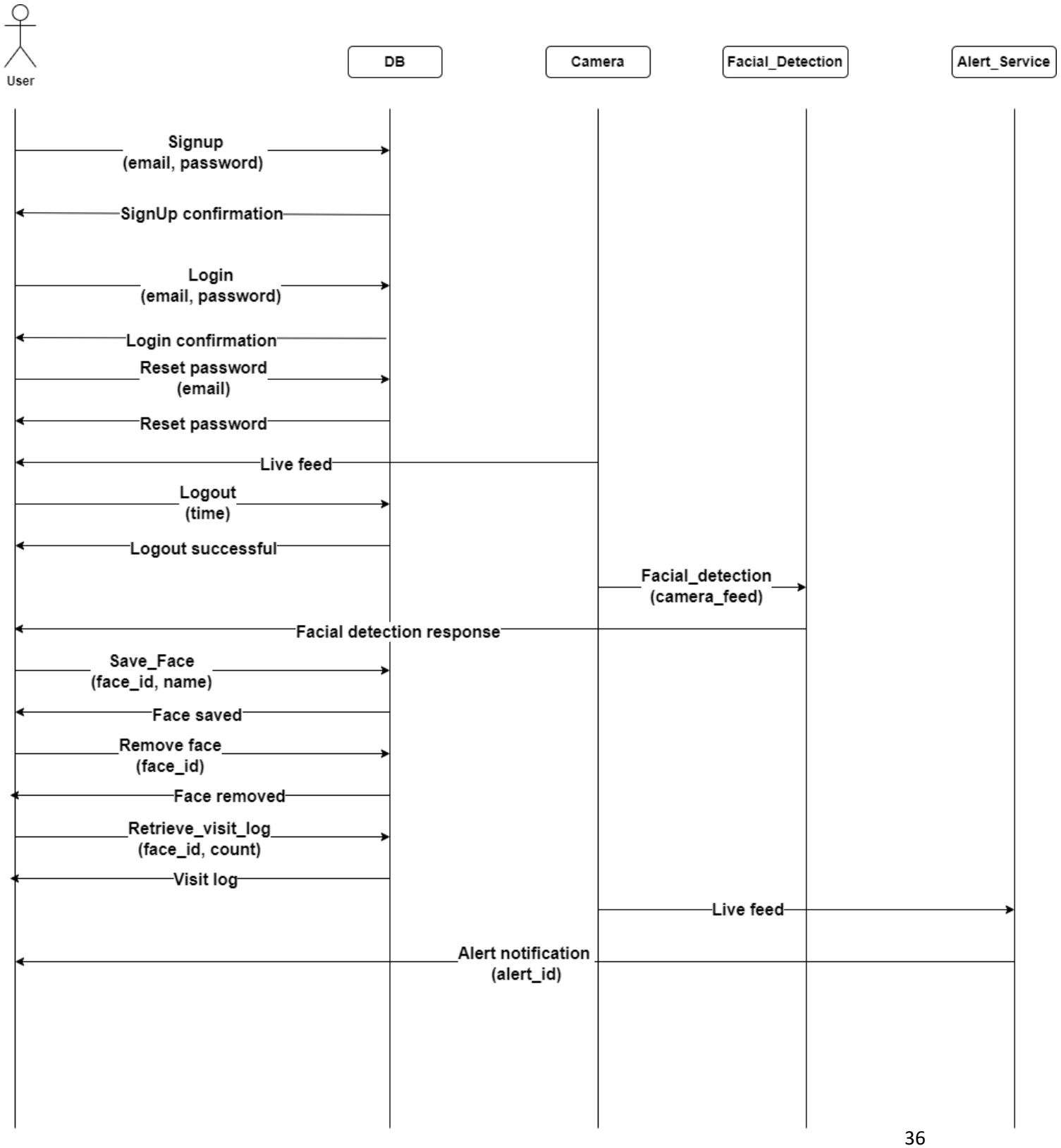


Figure 7. Class Diagram

3.4. Sequence Diagram



3.5. Entity Relationship Diagram

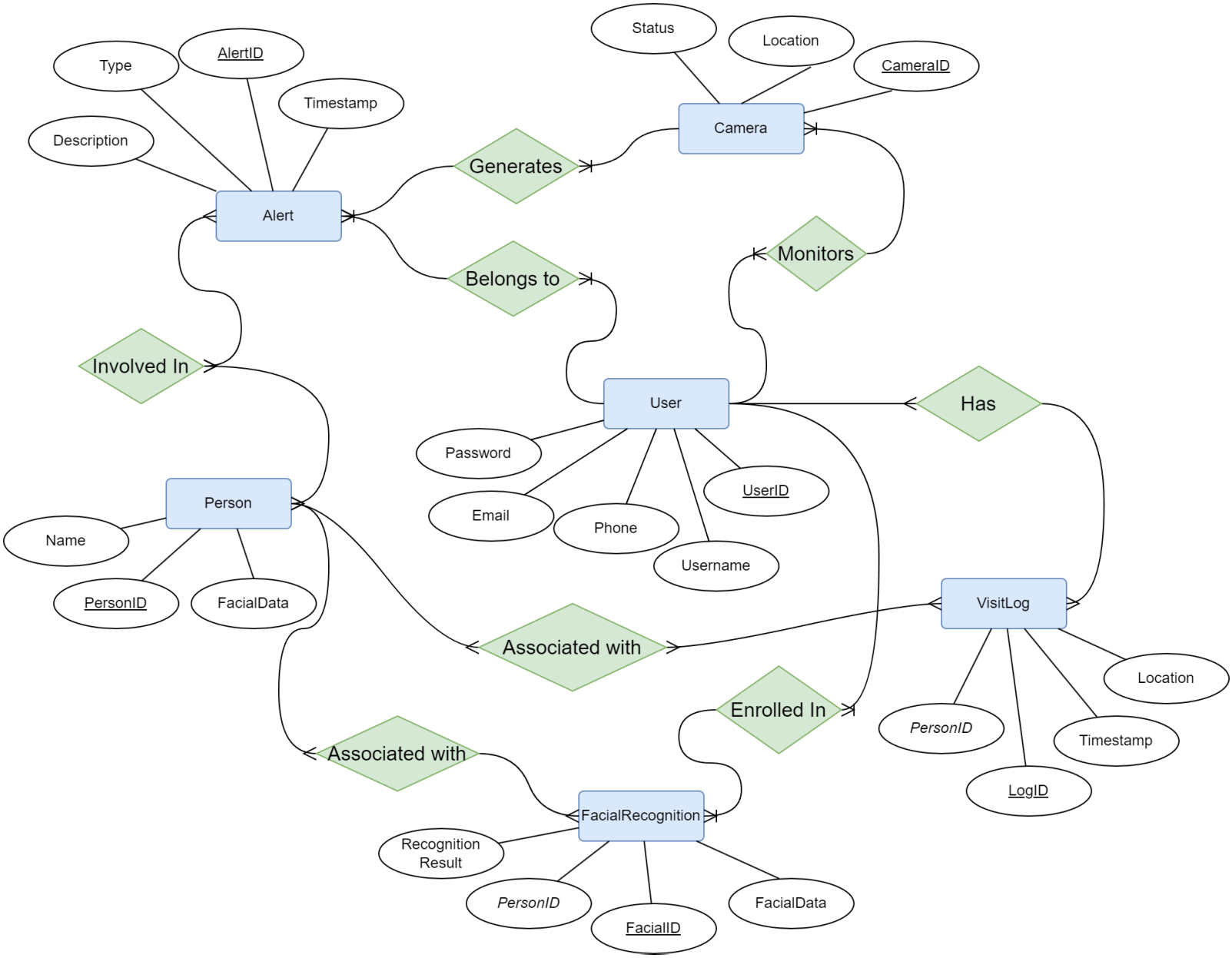


Figure 8. Entity Relationship Diagram

3.6. Database Schema

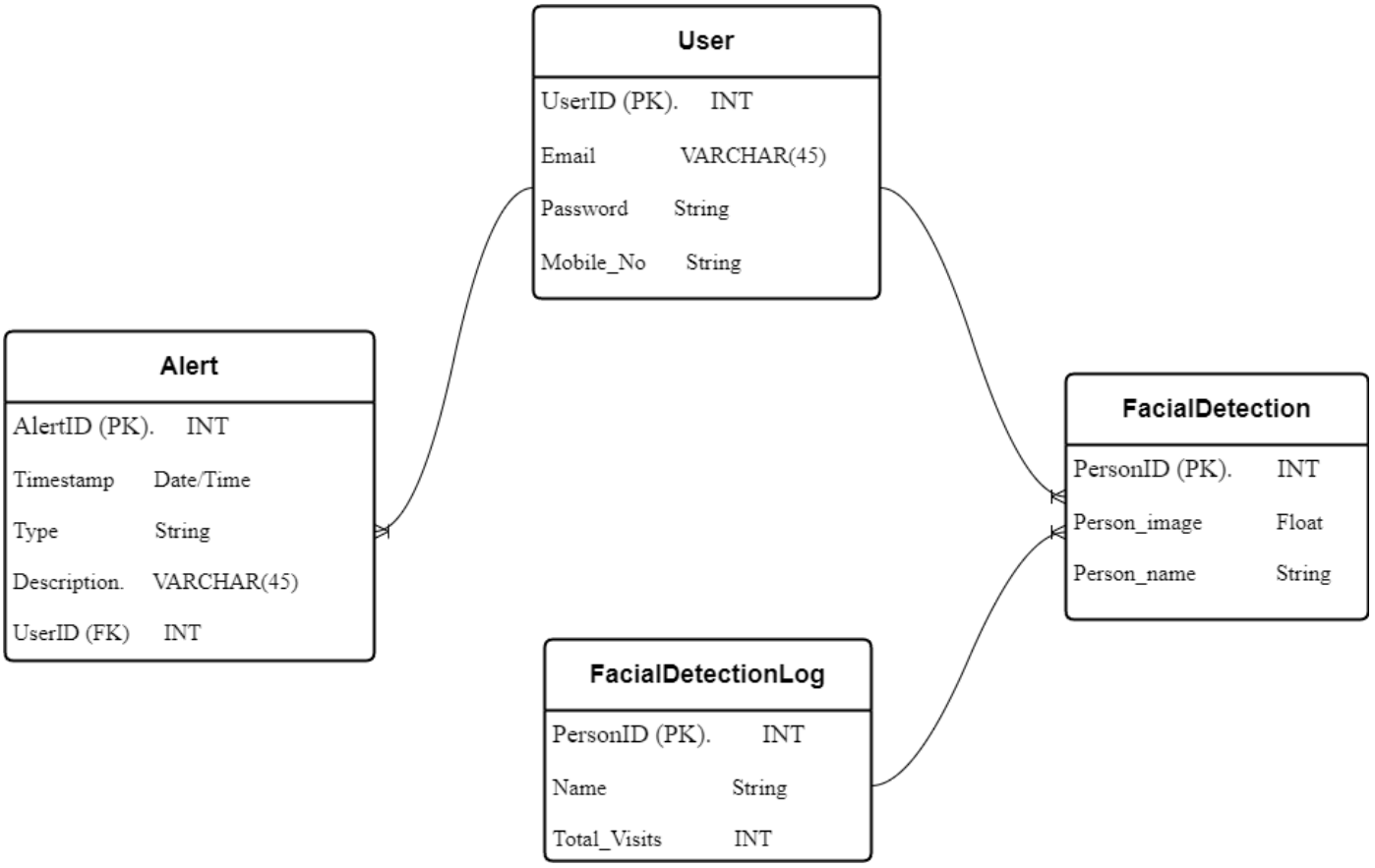


Figure 10. Database Schema

3.7. User Interface Design

UI Design is all about making things easy for people to use on the computer or phone. It's like guessing what users want to do and then making sure the buttons and stuff on the screen are easy to understand and use. UI Design brings together different ideas about how things should look and work, so it's not confusing for the people using it.

Sign Up:

A new user is required to register by accessing the signup page, where they will provide their email address and create a password.

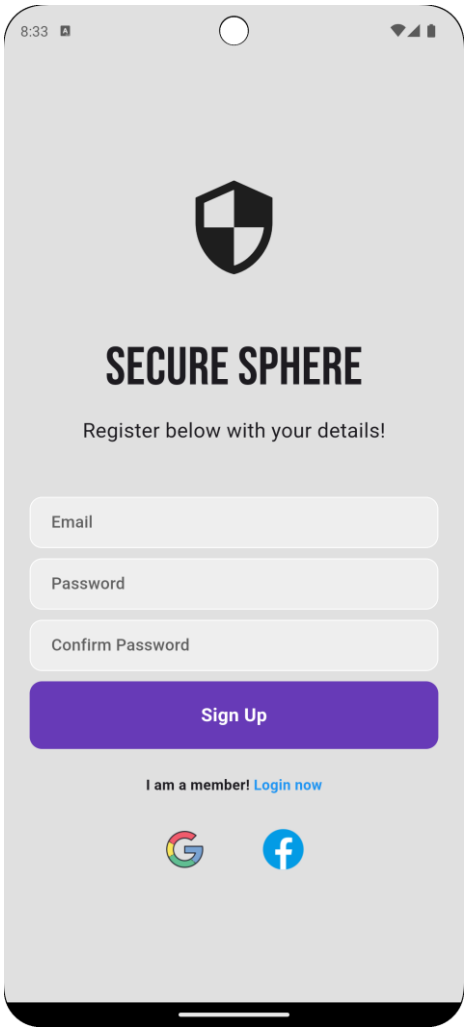


Figure 11. Signup Image

Login:

This is the user login interface where users input their email and password.

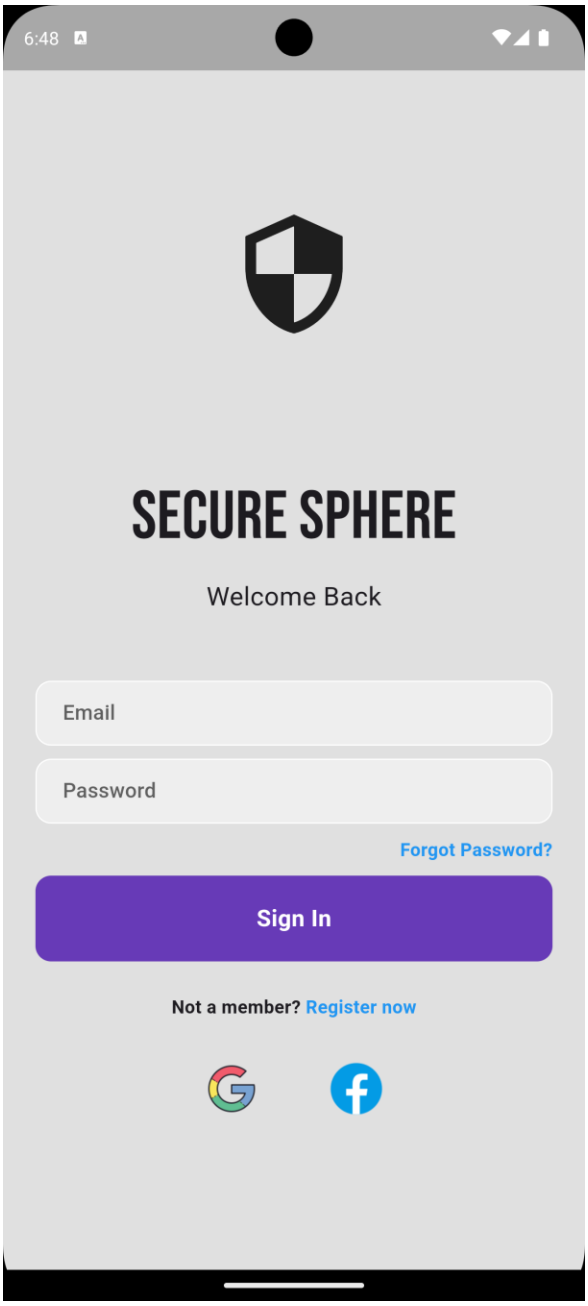


Figure 12. Login Image

Home Page:

This is the homepage where the live camera feed will be displayed.

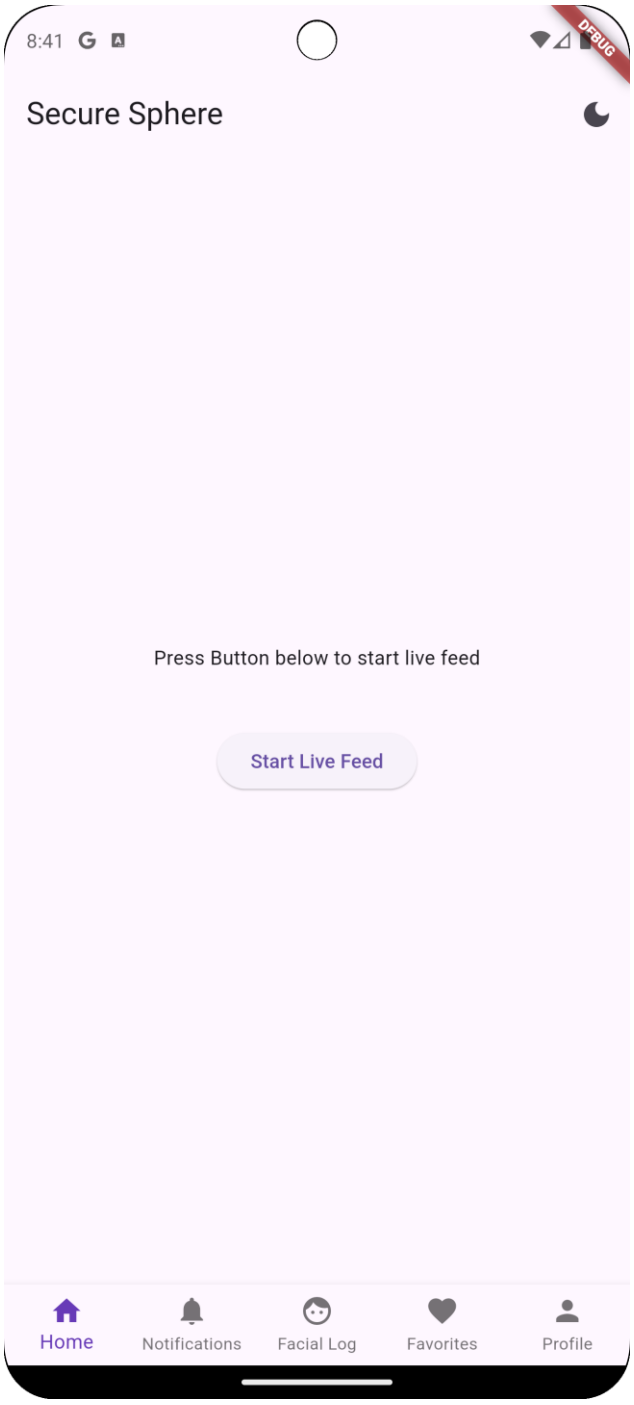


Figure 13. Home Page Image

Notification:

This is the notification page it will show different kinds of alert messages.

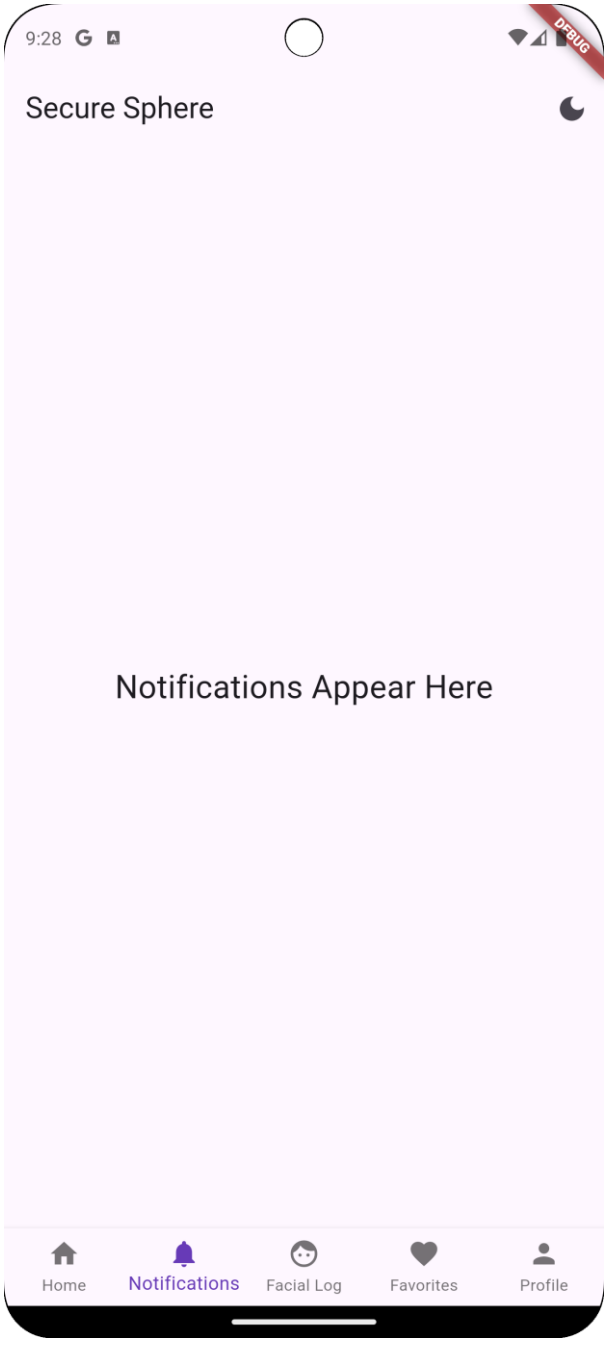


Figure 14. Notifications Image

Profile:

This is the profile page, showcasing user details including a profile picture, name, email, mobile number, gender, and a logout button.

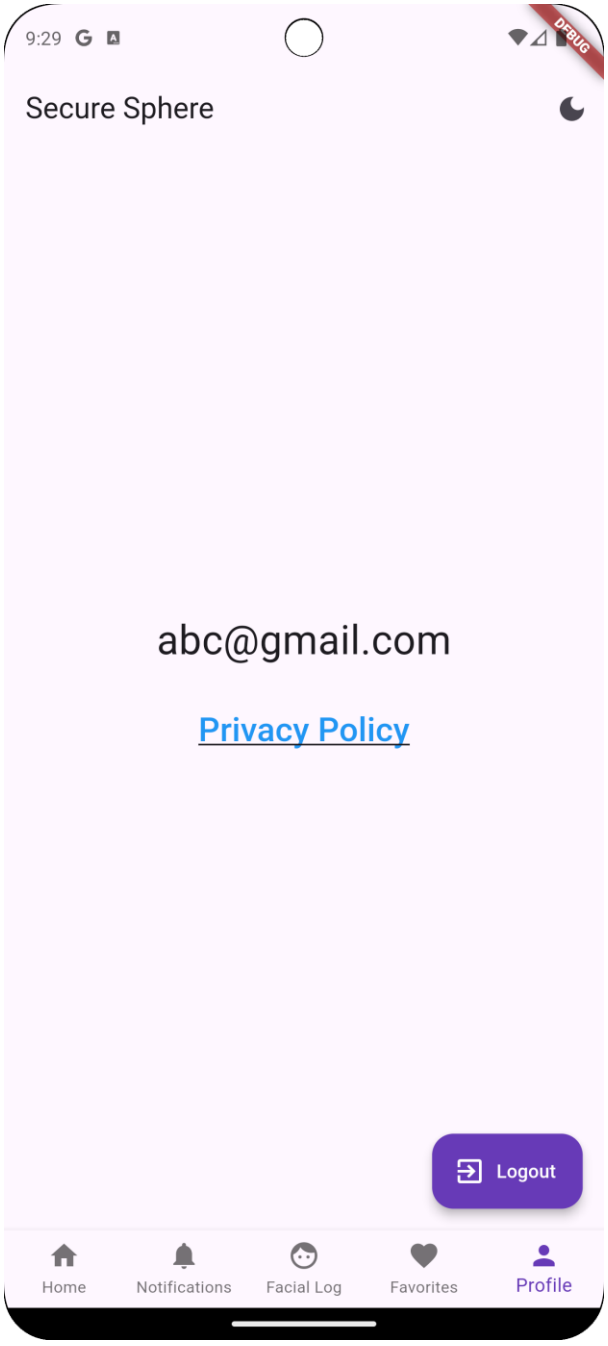


Figure 15. Profile Image

Facial Log:

This is facial log, it will show the detected/saved faces.

Figure 16. Facial Log Image

Favorites Facial Log:

This is show favorite facial logs saved by the user.

Figure 17. Favorites Facial Log

Chapter 4

Software Development

4.1 Coding Standards

The adopted coding standards are discussed in the following subsections.

4.1.1 Indentation

The code consistently uses two spaces as the unit of indentation, maintaining this pattern throughout the entire code.

4.1.2 Declaration

One declaration per line is used to enhance the clarity of code. The order and position of declaration is as follows:

I. **main.dart Entry Point:**
- main() function initializes Flutter and Firebase.
- Runs the MyApp widget.

II. **MyApp Class:**
- StatelessWidget representing the main application.
- Configures MaterialApp and sets MainPage as the initial screen.

III. **MainPage Class:**
- Uses StreamBuilder to determine user authentication status.
- Displays either HomePage or AuthPage based on authentication.

IV. **AuthPage Class:**
- StatefulWidget managing toggling between login and registration screens.
- Uses showLoginPage boolean to determine screen display.

V. **HomePage Class:**
- Displays user information and a "Sign Out" button if authenticated.

- Utilizes StreamBuilder to respond to authentication state changes.

VI. LoginPage Class:

- Stateful widget providing a login screen with email and password fields.
- Implements signIn method for Firebase Authentication.

VII. RegisterPage Class:

- Stateful widget offering a registration screen with email, password, and confirmation fields.
- Implements signUp method for user registration if passwords match.

VIII. ForgotPasswordPage Class:

- Allows users to request a password reset by entering their email.
- Implements passwordReset method using Firebase Authentication.

IX. Modular Design:

- Organized code into separate files for distinct components/screens.
- Ensures clear separation of concerns for better maintainability.

X. Firebase Authentication:

- Utilizes Firebase Authentication services for user authentication.
- Streamlines the user experience based on authentication state changes.

4.1.3 Statement Standards

The code maintains readability through clear spacing and an organized structure. Overall adherence to coding standards ensures a uniform and maintainable codebase, promoting effective collaboration. The code aligns with established Flutter practices, reflecting a standardized and coherent coding style.

4.1.4 Naming Convention

Naming conventions improve program readability by facilitating easier comprehension, making the code more accessible and understandable. Following are the conventions that were used:

We use clear and detailed English words to accurately explain what variables, methods, or classes all are about.

- Use terminology that aligns with the specific domain or context.
- Employed a mix of lowercase and uppercase letters for better readability.
- Capitalize the first letter of class names for consistency.

4.2 Development Environment

Flutter, from Google, is a free tool for making apps on different devices like phones and computers. It uses one code language (Dart) and has a lot of customizable parts (widgets) to make building apps fast and easy with a feature called hot reload. Flutter provides developers with various features such as:

- Write code once and deploy it on multiple platforms like iOS, Android, web, and desktop.
- Build expressive and flexible user interfaces using a rich set of customizable widgets.
- Instantly see the impact of code changes, facilitating rapid development and debugging.

These features streamline the development process, making app creation efficient and versatile

4.3 Database Management Firebase

Firebase has been implemented in your Flutter application, from where it will be able to perform CRUD operations.

4.4 Software Description

4.4.1 Firebase Authentication Sign in

```
Future signIn() async {
   await FirebaseAuth.instance.signInWithEmailAndPassword(
     email: _emailController.text.trim(),
     password: _passwordController.text.trim(),
   );
 }
```

Description:

This function uses Firebase Authentication to sign in a user with the provided email and password.

4.4.2 Gesture Detector for Forgot Password

```
GestureDetector(
            onTap: () {
              Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) {
                    return ForgotPasswordPage();
                  },
                ),
              );
            },
            child: Text(
              'Forgot Password?',
              style: TextStyle(
                color: Colors.blue,
                fontWeight: FontWeight.bold,
              ),
            ),
          ),
```

Description:

This GestureDetector triggers navigation to the 'ForgotPasswordPage' when the user taps on the "Forgot Password?" Text.

4.4.3 Login Navigation link

```
Row(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Text(
        'I am a member!',
        style: TextStyle(
          fontWeight: FontWeight.bold,
        ),
      ),
      GestureDetector(
        onTap: widget.showLoginPage,
        child: Text(
          ' Login now',
          style: TextStyle(
            color: Colors.blue,
            fontWeight: FontWeight.bold,
          ),
        ),
      ),
    ],
  ),
```

Description:

This UI component provides an option for users who are already members to navigate to the login page, enhancing user experience.

4.4.4 Password Confirmation Check

```
bool passwordConfirmed() {
    if (_passwordController.text.trim() ==
        _confirmpasswordController.text.trim()) {
      return true;
    } else {
      return false;
    }
  }
```

Description:

This function checks if the entered password matches the confirmed password, ensuring a secure user registration process.

4.4.5 Signing Out Button

```
MaterialButton(
      onPressed: () {
        FirebaseAuth.instance.signOut();
      },
      color: Colors.deepPurple[200],
      child: Text('sign out'),
    )
```

Description:

This MaterialButton triggers the sign-out process when pressed, using **FirebaseAuth.instance.signOut()**

Chapter 5

Software Testing

Software testing is crucial for the evaluation of software components and behavior of the software under certain circumstances. This section offers an explanation of the testing process used. It covers the testing approach, methodology, the suite of tests employed, and the outcomes of testing for the created software.

5.1. Testing Methodology

Having a testing plan is crucial to ensure a stable and reliable product delivery within a predictable timeframe. Our chosen testing methodology is manual software testing, specifically Black Box Testing. This method assesses software functionalities without knowledge of internal code details, focusing on input and output based on requirements.

5.2. Test Environment

- Windows 11
- Mac OS Sonoma
- Android Studio
- Xcode

5.3. Test Cases
5.3.1. Test Case 1

Table 5: Test Case 1

Date: 29-01-2024	**Tester**: Mubeen
System: Secure Sphere	**Test ID**: 1
Objective: Sign up	**Test Type**: Blackbox Testing
Version: 1	
Input: *First Name*: Mubeen *Last Name*: Yaqub *Email*: bcs203165@cust.pk *Password*: Abcd@1234	
Expected Result: Account Created	
Actual Result: Passed	

5.3.2. Test Case 2

Table 6: Test Case 2

Date: 29-01-2024	**Tester**: Mubeen
System: Secure Sphere	**Test ID**: 2
Objective: Sign up	**Test Type**: Blackbox Testing
Version: 1	
Input: *First Name*: Mubeen *Last Name*: Yaqub *Email*: mobeen?yaqub@gmail.com *Password*: Abcd@1234	
Expected Result: Invalid email error	
Actual Result: Failed	

5.3.3. Test Case 3

Table 7: Test Case 3

Date: 29-01-2024	**Tester**: Mubeen
System: Secure Sphere	**Test ID**: 3
Objective: Sign up	**Test Type**: Blackbox Testing
Version: 1	
Input: *First Name*: Mubeen *Last Name*: Yaqub *Email*: *Password*: Abcd@1234	
Expected Result: Email missing	
Actual Result: Failed	

5.3.4. Test Case 4

Table 8: Test Case 4

Date: 29-01-2024	**Tester**: Mubeen
System: Secure Sphere	**Test ID**: 4
Objective: Sign up	**Test Type**: Blackbox Testing
Version: 1	
Input: *First Name*: Mubeen *Last Name*: Yaqub *Email*: mobeenyaqub@gmail.com *Password*:	
Expected Result: Password missing	
Actual Result: Failed	

5.3.5. Test Case 5

Table 9: Test Case 5

Date: 29-01-2024	**Tester**: Mubeen
System: Secure Sphere	**Test ID**: 5
Objective: Sign up	**Test Type**: Blackbox Testing
Version: 1	
Input: *First Name*: *Last Name*: Yaqub *Email*: mobeenyaqub@gmail.com *Password*: Alpha@123	
Expected Result: First name missing	
Actual Result: Failed	

5.3.6. Test Case 6

Table 10: Test Case 6

Date: 29-01-2024	**Tester**: Mubeen
System: Secure Sphere	**Test ID**: 6
Objective: Sign up	**Test Type**: Blackbox Testing
Version: 1	
Input: *First Name*: Mubeen *Last Name*: *Email*: mobeenyaqub@gmail.com *Password*: Alpha@123	
Expected Result: Last name missing	
Actual Result: Failed	

5.3.7. Test Case 7

Table 11: Test Case 7

Date: 29-01-2024	**Tester**: Mubeen
System: Secure Sphere	**Test ID**: 7
Objective: Log in	**Test Type**: Blackbox Testing
Version: 1	
Input: *Email*: mobeenyaqub@gmail.com *Password*: Alpha@123	
Expected Result: Log in successful	
Actual Result: Passed	

5.3.8. Test Case 8

Table 12: Test Case 8

Date: 29-01-2024	**Tester**: Mubeen
System: Secure Sphere	**Test ID**: 8
Objective: Log in	**Test Type**: Blackbox Testing
Version: 1	
Input: *Email*: faizan@email.com *Password*: Beta@123	
Expected Result: Email not registered	
Actual Result: Failed	

5.3.9. Test Case 9

Table 13: Test Case 9

Date: 29-01-2024	**Tester**: Mubeen
System: Secure Sphere	**Test ID**: 9
Objective: Log in	**Test Type**: Blackbox Testing
Version: 1	
Input: *Email*: mobeenyaqub@gmail.com *Password*: Beta@123	
Expected Result: Invalid password	
Actual Result: Failed	

5.3.10. Test Case 10

Table 14: Test Case 10

Date: 29-01-2024	**Tester**: Mubeen
System: Secure Sphere	**Test ID**: 10
Objective: Log in	**Test Type**: Blackbox Testing
Version: 1	
Input: *Email*: *Password*: Beta@123	
Expected Result: Email missing	
Actual Result: Failed	

5.3.11. Test Case 11

Table 15: Test Case 11

Date: 29-01-2024	**Tester**: Mubeen
System: Secure Sphere	**Test ID**: 11
Objective: Log in	**Test Type**: Blackbox Testing
Version: 1	
Input: *Email*: mobeenyaqub@gmail.com *Password*:	
Expected Result: Password missing	
Actual Result: Failed	

5.3.12. Test Case 12

Table 16: Test Case 12

Date: 11-02-2024	**Tester**: Mubeen
System: Secure Sphere	**Test ID**: 12
Objective: Facial Detection	**Test Type**: Blackbox Testing
Version: 1	
Input: Live camera feed with face	
Expected Result: Face stored as image	
Actual Result: Passed	

5.3.13. Test Case 13

Table 17: Test Case 13

Date: 11-02-2024	**Tester**: Mubeen
System: Secure Sphere	**Test ID**: 13
Objective: Facial Detection	**Test Type**: Blackbox Testing
Version: 1	
Input: Live camera feed with no face	
Expected Result: No image saved	
Actual Result: Passed	

5.3.14. Test Case 14

Table 18: Test Case 14

Date: 17-07-2024	**Tester**: Mubeen
System: Secure Sphere	**Test ID**: 14
Objective: Strolling Alerts	**Test Type**: Blackbox Testing
Version: 1	
Input: Live camera feed with person detected for 30 seconds	
Expected Result: Notification generated	
Actual Result: Passed	

5.3.15. Test Case 15

Table 19: Test Case 15

Date: 17-07-2024	**Tester**: Mubeen
System: Secure Sphere	**Test ID**: 15
Objective: Strolling Alerts	**Test Type**: Blackbox Testing
Version: 1	
Input: Live camera feed with person detected for less than 30 seconds	
Expected Result: No notification generated	
Actual Result: Passed	

5.3.16. Test Case 16

Table 20: Test Case 16

Date: 17-07-2024	**Tester**: Mubeen
System: Secure Sphere	**Test ID**: 16
Objective: Trespassing Alerts	**Test Type**: Blackbox Testing
Version: 1	
Input: Live camera feed with a person of unknown face detected inside premises	
Expected Result: Notification generated	
Actual Result: Passed	

Chapter 6

Software Deployment

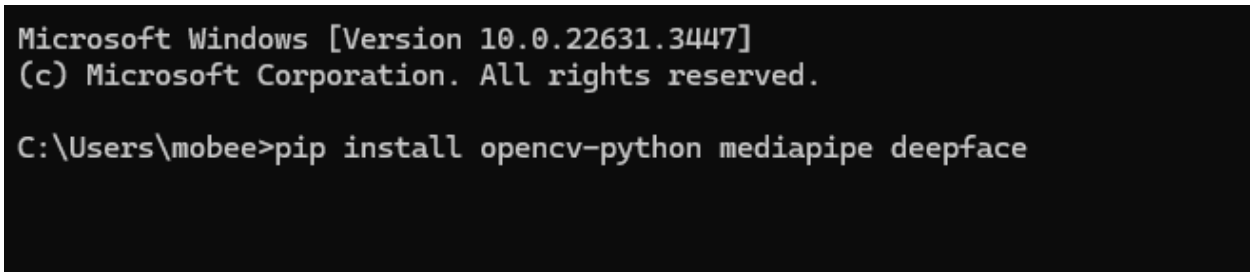
6.1 Installation / Deployment Process Description:

- In order to run the computer vision model one has to import the following Python modules/packages/libraries.
  1. opencv-python
  2. os
  3. datetime
  4. mediapipe
  5. concurrent.futures
  6. deepface

  To install these packages a user has to run this command

  pip install opencv-python mediapipe deepface

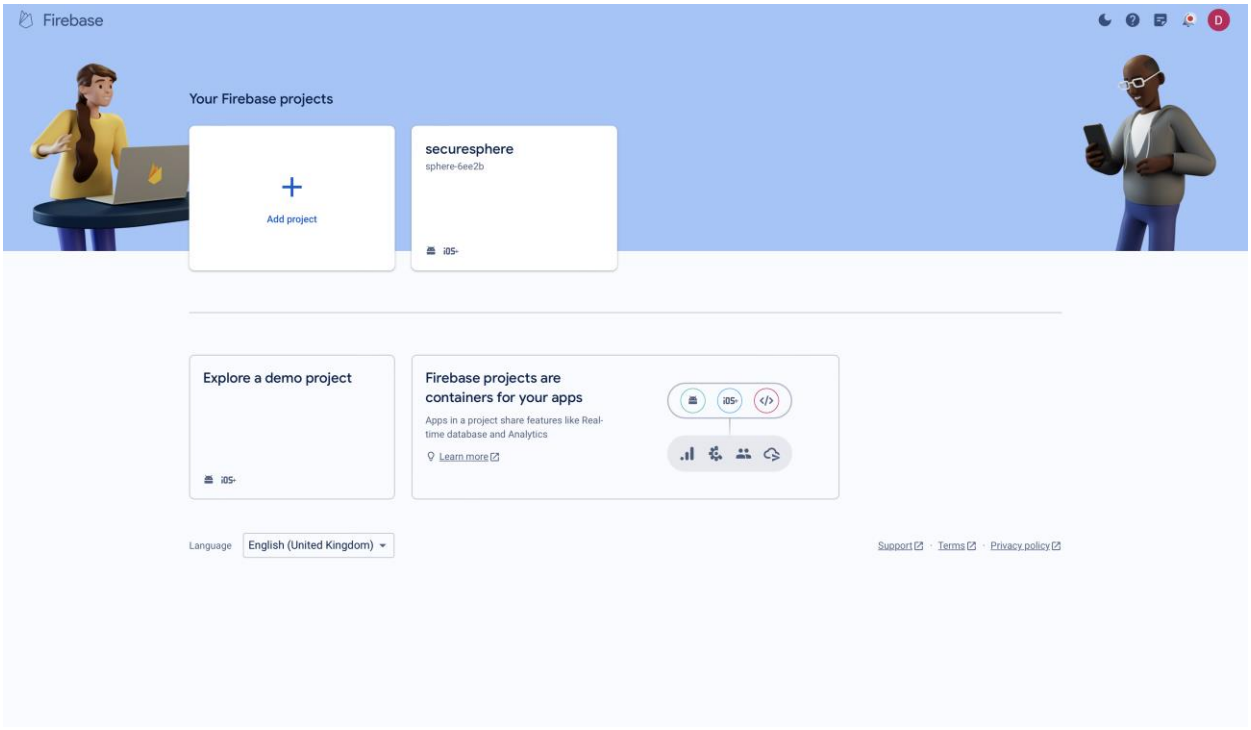
  The remaining three packages come built-in



6.2 Firebase Configuration Process:

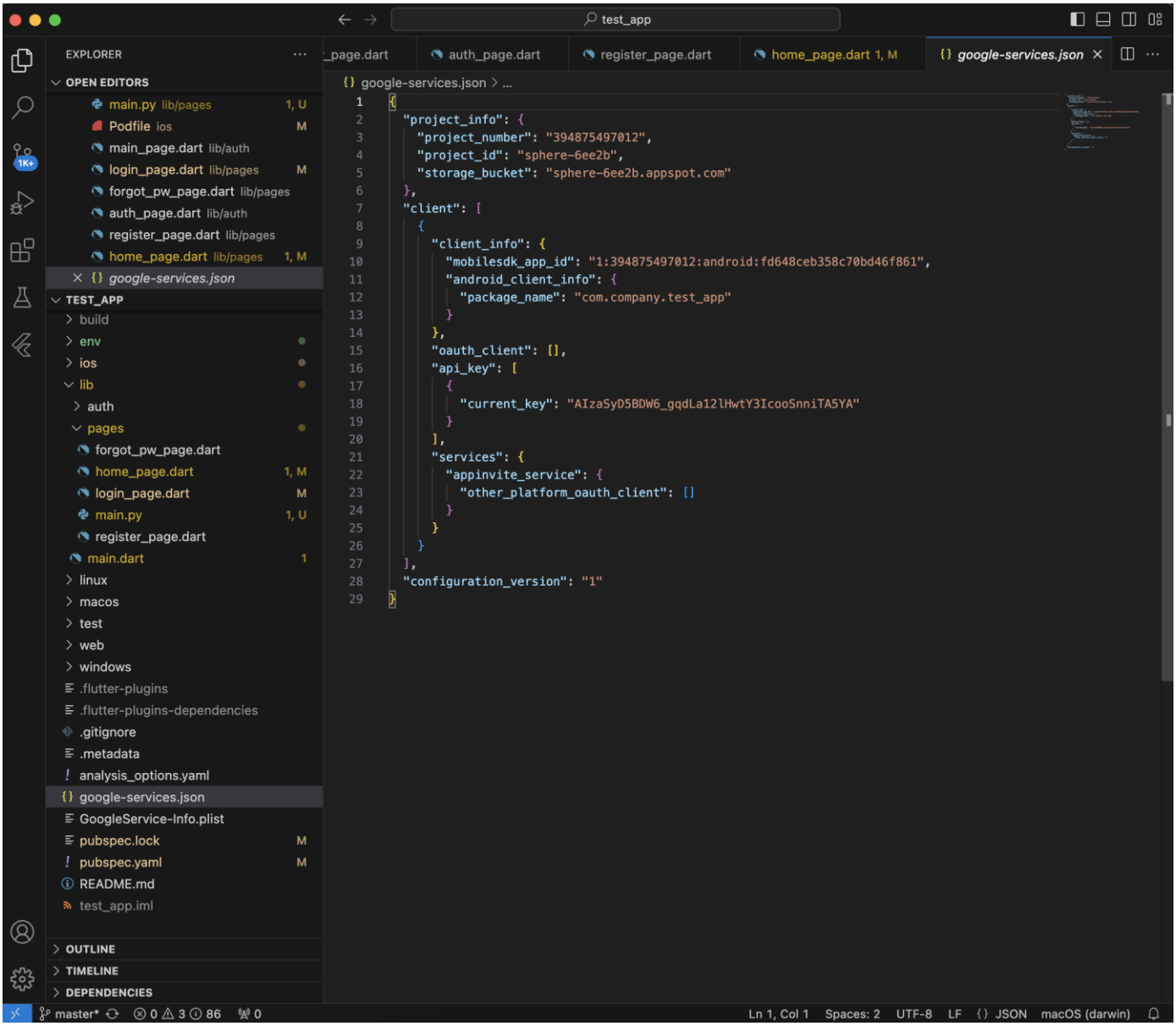
Step 1: Create a Firebase project

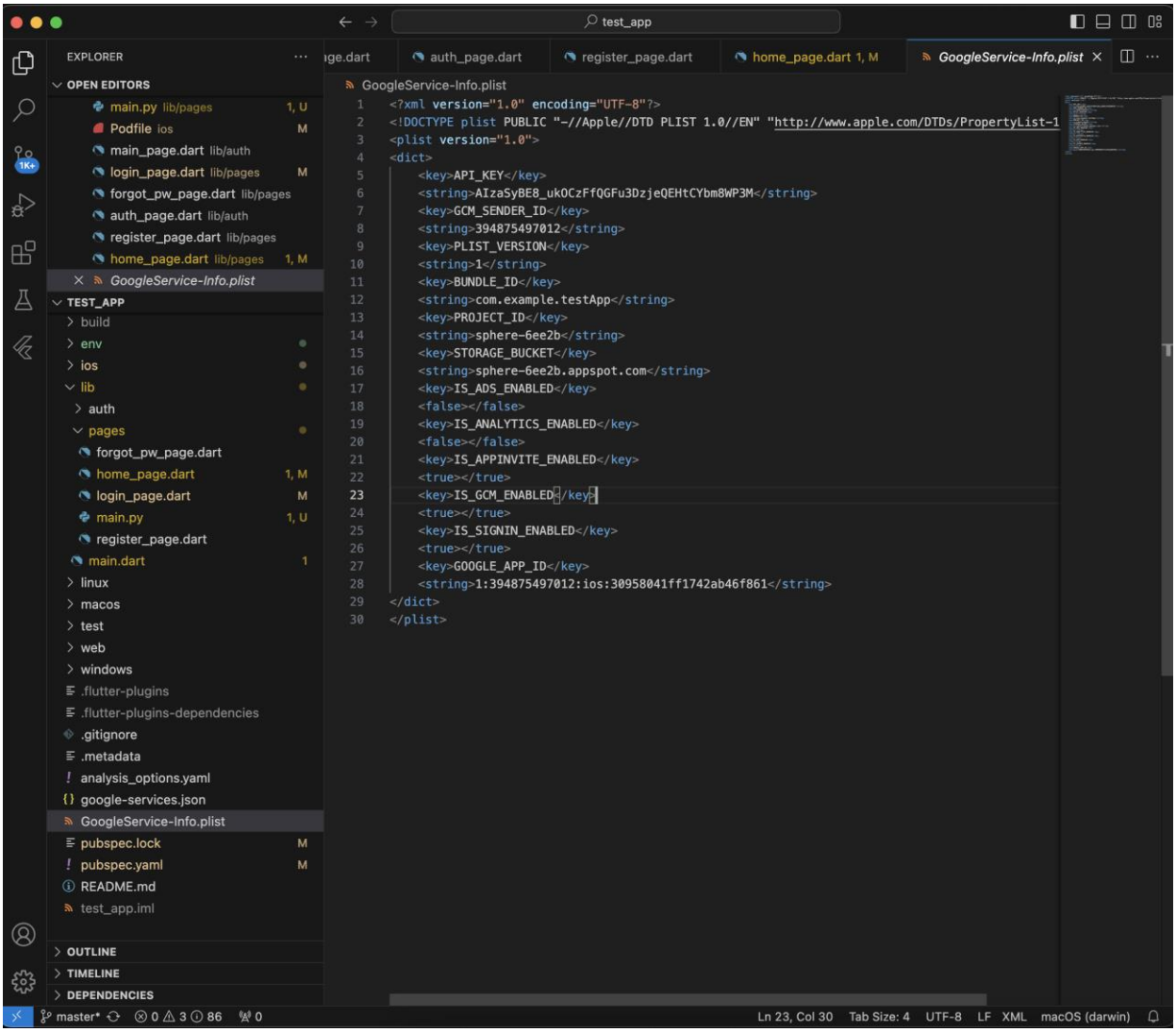
1. Navigate to the Firebase console (https://console.Firebase.google.com/)
2. Click on “Add project” and create a new project



Step 2: Register your app with Firebase

1. After creating the project click on ‘IOS’ or ‘Android’ to add the app to Firebase project
2. Enter your app's package name (e.g., com.example.app) and other optional details.
3. Download the configuration file (google-services.json for Android, GoogleService-Info.plist for iOS) and place it in the project directory.


test_app
EXPLORER
OPEN EDITORS
main.py lib/pages 1, U
Podfile ios M
main_page.dart lib/auth
login_page.dart lib/pages M
forgot_pw_page.dart lib/pages
auth_page.dart lib/auth
register_page.dart lib/pages
home_page.dart lib/pages 1, M
google-services.json
TEST_APP
build
env
ios
lib
auth
pages
forgot_pw_page.dart
home_page.dart 1, M
login_page.dart M
main.py 1, U
register_page.dart
main.dart 1
linux
macos
test
web
windows
.flutter-plugins
.flutter-plugins-dependencies
.gitignore
.metadata
analysis_options.yaml
google-services.json
GoogleService-Info.plist
pubspec.lock M
pubspec.yaml M
README.md
test_app.iml
OUTLINE
TIMELINE
DEPENDENCIES
_page.dart
auth_page.dart
register_page.dart
home_page.dart 1, M
google-services.json
{ } google-services.json > ...
{
"project_info": {
"project_number": "394875497012",
"project_id": "sphere-6ee2b",
"storage_bucket": "sphere-6ee2b.appspot.com"
},
"client": [
{
"client_info": {
"mobilesdk_app_id": "1:394875497012:android:fd648ceb358c70bd46f861",
"android_client_info": {
"package_name": "com.company.test_app"
}
},
"oauth_client": [],
"api_key": [
{
"current_key": "AIzaSyD5BDW6_gqdLa12lHwtY3IcooSnniTA5YA"
}
],
"services": {
"appinvite_service": {
"other_platform_oauth_client": []
}
}
}
],
"configuration_version": "1"
}
master*
0 3 86
0
Ln 1, Col 1
Spaces: 2
UTF-8
LF
{ } JSON
macOS (darwin)



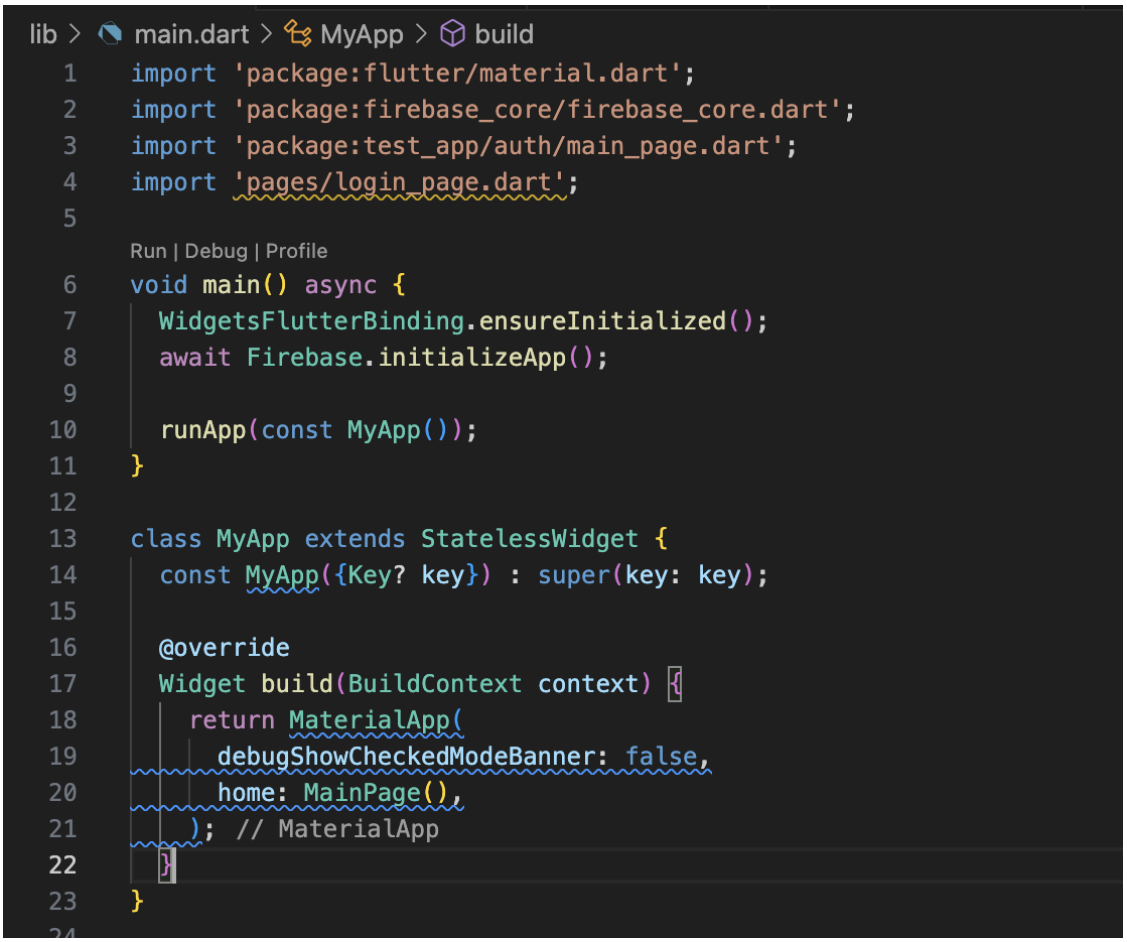
Step 3: Add Firebase SDK Dependencies

1. Open flutter project's 'pubspec.yaml' file
2. Add Firebase core and authentication dependencies

```
dependencies:
  flutter:
    sdk: flutter
  firebase_core: ^2.28.0
  firebase_auth: ^4.19.0
```

Step 4: Initialize Firebase in flutter app

1. Initialize Firebase in main() function
2. In authentication-related files (login_page.dart, register_page.dart, etc.), we implemented the same autAuhentication logic with Firebase Authentication methods.



6.3 Play Store Deployment Details:

1. APK file (.apk)
2. App bundle file (.aab)
3. Key Store (.jsk)
4. Description of application
5. Application icon
6. Privacy Policy
7. Deletion Policy
8. Application screenshots
9. Age rating
10. Application category and sub-category
11. Application source code

Download link:
https://play.google.com/store/apps/details?id=com.company.test_app

Chapter 7

Project Evaluation

The report of the project “Secure Sphere” has been approved based on the following evaluation guidelines:

7.1. Project Evaluation Report

Ms. Tayyaba Zaheer	Dr. M. Abdul Qadir	Mr. Muhammad Zulqarnain
(Supervisor)	(Examiner 1)	(Examiner 2)

References

Deepface framework: https://github.com/serengil/deepface