MULTIMEDIA UNIVERSITY

FACULTY OF ENGINEERING

FINAL REPORT

EPE3016 CAPSTONE PROJECT

EC02

TRIMESTER 1 SESSION 2022/2023

GROUP No.: 02

Declaration of originality:

I declare that all sentences, results and data mentioned in this report are from my own work. All work derived from other authors have been listed in the references. I understand that failure to do this is considered plagiarism and will be penalized.

Note that copying and any act of cheating in the report, results and data are strictly prohibited.

Student Name	Student ID	Major	Task	Signature
MARAWAN ASHRAF FAWZY AHMED ELDEIB	**1181102334**	**CE**	**Mobile Application**	**MARAWAN**
ALJUHANI, HANEEN RADI	1181303150	LE	Sensor System	HANEEN
AHMED MOHAMMED AHMED KHUDHAIR	1191302289	LE	Actuator System	AHMED

Abstract

In this project, we present an IOT based smart door lock system which aims to enhance home security by integrating various sensors and technologies to provide a low-cost, user-friendly, and secure solution. The system features a power-saving mechanism that enables it to conserve energy by detecting motion using PIR sensors and confirming the proximity with ultrasonic sensors. The system then wakes up the fingerprint sensor for verification and unlocking the door lock. In case of a failed verification, the system triggers an alarm buzzer. The system also connects to the internet and sends notifications to the user through a real-time firebase based mobile application. The application allows the user to monitor and control the door lock, track motion and entry history, set up new fingerprints, and call for emergency assistance if needed.

Table of Contents

List of Figures

List of Tables

List of Abbreviations (optional)

IOT – Internet of Things

PIR – Passive infrared

App – Application

Add - Addition

Chapter 1: Introduction

1.1 Introduction

IOT, or the Internet of Things, refers to the network of physical devices that are embedded with electronics, software, and sensors, enabling them to connect and exchange data with each other.

In today's fast-paced world, the security and safety of our homes have become a top priority. The growing need for enhanced home security has prompted the development of smart lock systems. Conventional locks are now being replaced by smart locks that offer greater convenience and improved security. The IOT based smart door lock system provides a convenient, secure, and low-cost solution for home security and monitoring. The system incorporates various sensors, a microcontroller, and a mobile application to provide real-time access and control of the lock.

1.2 Problem statement

With the increasing number of home invasions and theft incidents, ensuring the security of our homes has become a major concern. Homeowners are often not aware of the security risks they face from intruders and robbers. Locking the doors is not enough to ensure home security, traditional lock systems are not equipped to provide real-time monitoring and notification, leaving homeowners vulnerable to intrusion and theft. Homeowners need to be informed about any security breaches immediately. This project aims to provide a solution that is both low-cost and user-friendly, while ensuring the safety and security of the homeowner's belongings.

1.3 Motivation and objectives

Motivation:

Despite the availability of various home security systems in the market, they are too expensive too complicated to use. Hence, the primary motivation behind this project

is to provide a low-cost, reliable, power saving and user-friendly home security system.

Objectives:

- To design and develop a low-cost and user-friendly home security system that provides real-time monitoring and control of the lock and motion through a mobile application.
- To incorporate a biometric system, fingerprint lock, to enhance the security of the system.
- To provide real-time notifications to the homeowner through a mobile application, including the ability to call emergency services if needed.
- To provide remote access to the lock through the mobile application, allowing the homeowner to unlock the door remotely.
- To provide real-time access to entry history, allowing the homeowner to view when the door was unlocked and by whom.
- To allow the homeowner to set up multiple fingerprints.
- To provide a secure communication system between the system and the mobile application, ensuring that the homeowner's data and privacy are protected.
- To provide a user-friendly and intuitive mobile application that is easy to use, even for those with limited technical knowledge.
- To eliminate the need for a monthly fee by providing a low-cost prototype of the home security system.
- To utilize power-saving technologies that reduce the energy consumption of the system, making it more environmentally friendly.
- To implement a motion detection system that wakes up the system when motion is detected and puts it to sleep when there is no movement, reducing power consumption and increasing battery life.
- To incorporate a buzzer that will alert the homeowner if the system detects a potential threat, such as an unauthorized person trying to enter the home.

1.4 Overview of report

In our project, we'll make use of the Internet of Things(IoT) great ability to monitor, detect, and respond with a microcontroller to give a house an extra layer of security. To give homeowners a greater sense of security by enabling them to use their smartphones to detect any motion outside their homes that might indicate burglars.

We'll create a system “IOT based smart door lock system” that monitors time for each activity, notifies the owner, and operates in response to the owner's choices. In order to conserve energy and ensure that only authorised residents can enter by unlocking the door, the home is also guarded by a fingerprint sensor that only activates when movement is detected. A PIR sensor and an ultrasonic sensor are used to monitor the unexpected movements. The system also includes a mobile application that receives alerts from the PIR and ultrasonic sensors and then notifies the user through cell phone as well as the nearby via buzzer. The Firebase IoT platform, which is linked to the circuit using an ESP8266 module, will be used to continuously monitor the system. The report provides an overview of the project, the design and implementation of the system, and the results of the tests performed.

Chapter 2: Review of Literature

First Literature:

2.1.1 Review of existing implementation techniques

A, Ramkumar, T. Vaigaiselvam, S. Rajendran, S. Saravanavel, A. Kamalesh and K. Rajesh, "Android Controlled Smart Home Automation with Security System," 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), 2022, pp. 1245-1249, doi: 10.1109/ICACITE53722.2022.9823575.

Problem that the author is addressing:
If you need immediate assistance when alone at home or while your home is being attacked.

Methods used:
Application can manage any appliance in your house and the special function is Ultra Panic Mode. First screen is login verification and then can control different devices. The user's information will be sent to the neighbourhood and local police stations when the panic mode is "Enabled." User information such as location and profile information

Results:
Using a smartphone application to turn things like fans and lights on and off and to message with the users details like name and location to the police in an emergency.

2.1.2 Summary on their advantages/disadvantages

Advantages

- User-controlled and monitored system.
- Remote monitoring capability.
- Panic mode for emergency situations

Disadvantages

- No alert system to notify the user or nearby individuals of suspicious motion.
- The system relies on an Arduino GSM module, which requires a sim card to connect to the internet and cannot be connected to home Wi-Fi like the ESP8266.
- No records of user activities or history.

Second Literature

2.2.1 Review of existing implementation techniques

S. Lokesh, S. B. Patil and A. Gugawad, "Home Security And Automation Using NodeMCU-ESP8266," 2020 IEEE Bangalore Humanitarian Technology Conference (B-HTC), 2020, pp. 1-6, doi: 10.1109/B-HTC50970.2020.9297917

Problem that the author is addressing:
Users must physically interact with door locks in order to unlock them, hence the current configuration of existing home security systems, such as digital locks, mechanical locks, etc., follows a straightforward scenario. The current system appears to be fairly straightforward and easy to use, but it lacks internet and database connectivity.

Theories/methods used:
The biometric Fingerprint sensor is going to be used to determine and recognize the person who is trying to enter the house, if the fingerprint matches the fingerprint in the system, the solenoid lock will be opened, otherwise it will stay closed. All of this process is happening using NodeMCU ESP8266, which sends and update the information into the cloud, and then sent to the application.

RESULTS:
The results are shown in two ways, in the cloud which shows the status of the door, and when the last time the status was changed. In the application, it shows if the lock and other devices are on or off.

2.2.2 Summary on their advantages/disadvantages

Advantages:

- The reliability of the system as the user won't be able to enter the house until the entered fingerprint is matching with one of the fingerprints saved in the system.
- Remotely controlled system.

Disadvantages

- The continuous operation of the fingerprint sensor uses more energy and shortens battery life.
- There is no records of the users.

Third Literature

2.3.1 Review of existing implementation techniques

S. Sinha, E. H. Teli and W. Tasnin, "Remote Monitoring and Home Security System," 2021 Innovations in Power and Advanced Computing Technologies (i-PACT), 2021, pp. 1-8, doi: 10.1109/i-PACT52855.2021.9696996.

Problem that the author is addressing:

How can someone monitor their house if they forgot to lock their house before going for work?

Theories/methods/models used:

Design a home security system that will sound a buzzer to alert the neighbours right away if it notices any strange movement. A PIR sensor is used to keep track of the odd motions. The system also includes a mobile application, which receives notifications from the PIR sensor and warns the user via his or her phone. The circuit is connected to the ESP8266 module-based ThingSpeak IoT platform, which can be used to continuously monitor the system.

RESULTS:

When a resident is not at home, the mobile application dashboard screen displays the phrase "HOUSE IS SAFE" informing him that his home is secure.

The buzzer sounds and sends a warning signal when the PIR sensor detects an unexpected movement, and the LED indicator lights to alert the user to the activity.

By alerting about the break-in through pop-up and push notification with the word "BREAK IN ALERT," the mobile application also reports the status of the home.

2.3.2 Summary on their advantages/disadvantages

Advantages:

- Alert system App indicates the status of the house like "BREAK IN" with the help of a PIR sensor which is used to track the odd motions and buzzer to make sound.
- continual remote home monitoring system to let residents keep an eye on suspicious activity when they're away from home.

Disadvantages:

- help of a PIR sensor which is used to track the odd motions and buzzer to make sound.

Chapter 3: Details of the Design

3.1 Design consideration (societal, health, safety, legal and cultural issues)

In our project, safety was a major design consideration. The safety of the homeowner and their family was our top priority. This is why we chose to incorporate a fingerprint sensor as the main means of unlocking the door. This ensures that only authorized residents can enter the home, minimizing the risk of intrusion by unauthorized individuals.

Additionally, we considered societal and cultural issues by making the system user-friendly and accessible to everyone, regardless of their technical knowledge. This was achieved by using a simple and intuitive mobile application that allows homeowners to easily monitor and control their home security system.

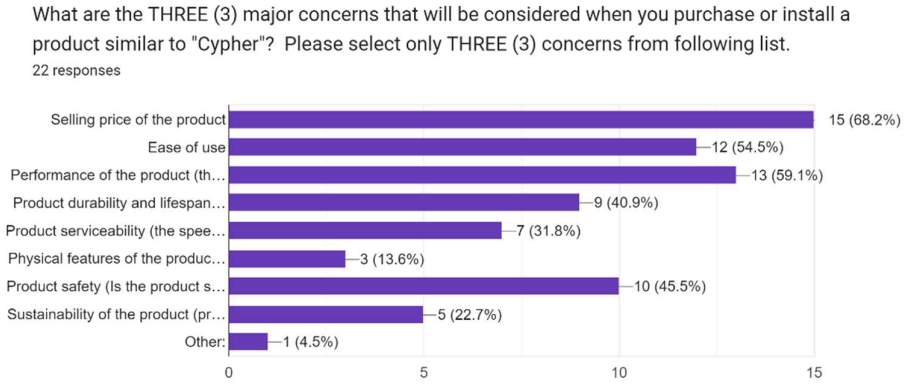

Figure 1 customer need analysis form

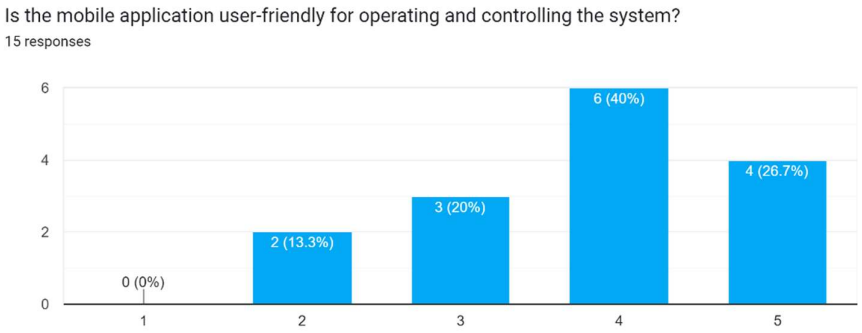

Figure 2 customer feedback analysis 1

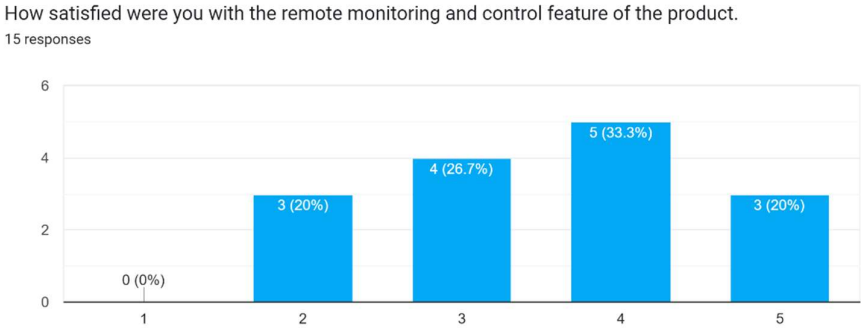

Figure 3 customer feedback analysis 2

The design of the mobile application software part of the project was created with various considerations in mind. The customer needs were taken into account to ensure that the solution provided was fulfilling the requirements of the target audience. The impact of the solution on society was also evaluated to make sure that the application would be of benefit to the end users. The software development process was carried out using legal software tools to ensure compliance with rules and regulations. This helped to ensure that the design process was carried out within the bounds of the law and no hazardous substances were used.

The design of the mobile application was created with cultural considerations in mind. The appearance and presentation of the app was designed to fit into the culture of the

target audience. The name, shape, and colour of the app were chosen carefully to ensure that they would not have any negative connotations in the target market.

A trade-off between various factors such as performance, power consumption, cost, ease of implementation, safety, and sustainability was considered in the design process. The aim was to find a balance between these factors to create a solution that was cost-effective, efficient, and user-friendly while also ensuring that safety and sustainability standards were met.

Moreover, health was also taken into consideration by ensuring that the system operates quietly, with minimal noise pollution. This is important, especially for homeowners who are sensitive to noise and need a quiet living environment.

Finally, the design of the mobile application software part of the project was carried out with consideration for societal, health, safety, legal, and cultural issues to provide the best possible solution for the end users. we considered legal issues by ensuring that the system complies with all relevant data privacy laws and regulations. The data collected by the system is encrypted and securely stored, protecting the homeowner's personal information and privacy.

3.2 Design configuration and analysis(software)

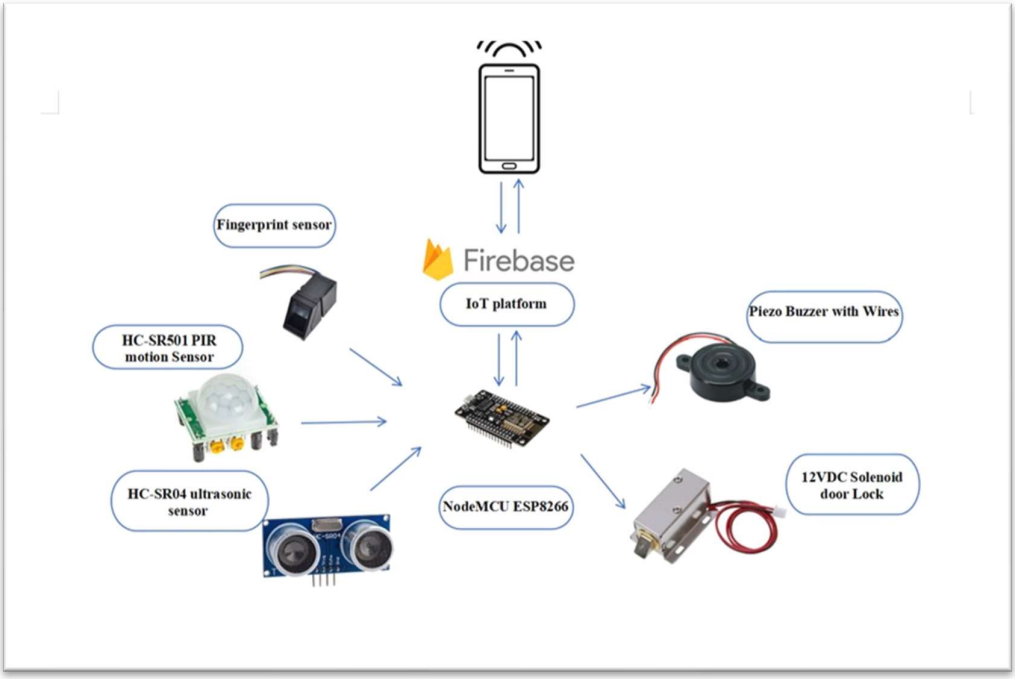

Figure 4 Overview of project components without battery

A new configuration called 'IOT based Smart door lock system' was designed to utilizes an ESP8266 module. The system includes a mobile application for remote control and monitoring, a biometric fingerprint sensor for secure entry, user friendly setup fingerprint through application and a PIR sensor for detecting any unexpected movements. The ESP8266 module will be connected to the Firebase IOT platform for continuous monitoring and will be able to send notifications to the user's phone in case of suspicious activity. The mobile application will also allow the user to keep a record of the users. Power saving mechanism is also implemented. To ensure that the system falls within the allocated budget, low-cost components such as ESP8266 and PIR sensor will be used.

IOT Cloud Firebase Realtime Database:

Figure 5 real-time firebase structure screenshot

Firebase Nodes:

distance: displays the ultrasonic distance (in cm) received from the NodeMCU esp8266.

doorLocked: displays the status of the door lock and allows the user to lock and unlock the door from the app.

motion: displays the status of motion detection and sends notifications to the user if motion is detected. And makes the distance display also.

option: by default, it is set to 2, but when the user wants to add a fingerprint, the app updates this node to 1 to set up a new fingerprint. When the setup is complete, the node is updated back to 2. 1 is for scanning a new fingerprint and 2 is for verifying the fingerprint to open the door lock.

Id: the ID that will be sent for the fingerprint sensor when entered to save the fingerprint on it.

profiles: consists of all home profiles with a different profile key for each profile. Each profile has a name and a list of fingerprints.

Fingerprints under profilekeys: each fingerprint ID is saved under different fingerkeys. When a new fingerprint is added, it is updated in the fingerprints list under that profile and it is also updated in the node 'Id' to send it to fingerprint sensor , so it is easier and faster for to identify update.

savedID: displays the fingerprint of the user who entered the home. After some time, it is updated back to zero. The app takes the ID and searches for it among all fingerprints under all profiles to find out who came home and returns the profile name and the date and time to 'history' node.

history: saves a list of data entries for different profiles with the data time when the fingerprint ID is sent to "savedID".

steps: for the fingerprint stepper widget, it updates the node with 1, then 2, then 3, then either 4 or 5 to guide the user through the fingerprint setup process. It informs the user if the setup was successful or failed at the end.or the fingerprint stepper widget, it updates node with 1 then 2 then 3 then either 4 or 5 to guide user to setup fingerprint and so at the end tells him if success or failed.

Android Application:

I developed a mobile application using Flutter, a popular open-source UI software development kit created by Google. Initially, I experimented with Kotlin but found Flutter to be more user-friendly and flexible. Flutter offers a range of easy-to-use options and simplifies the process of building cross-platform applications for various platforms such as Android, iOS, Linux, macOS, Windows, Google Fuchsia, and the web from a single codebase. This made it the ideal choice for my project. Overall, I believe Flutter is a valuable tool for developers due to its versatility and ease of use.

The application I built consists of eight pages in total, including a login page. Four of the pages serve as the main pages of the app: the monitor page, profile page, history page, and settings page. The remaining three pages are for specific functionalities, such as the fingerprint page, setup fingerprint page, and Wi-Fi connection page. These pages provide users with a comprehensive and user-friendly experience, making it easier for them to access the various features and functionalities of the app. The app also send notifications to user when motion is detected.

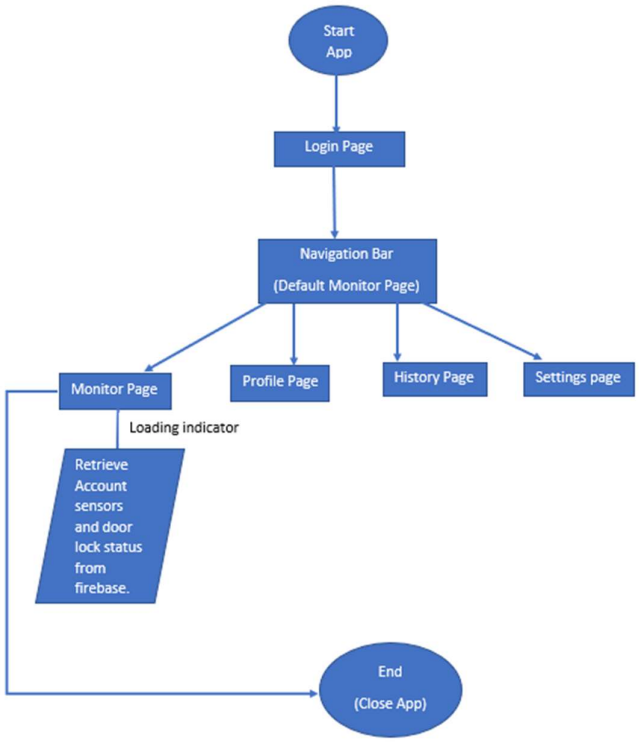

Figure 6 Basic main flow chart of app

Notification:

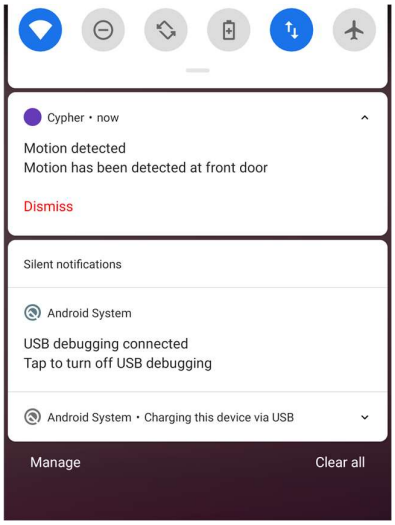

Figure 7 app motion notification alert screenshot

Login page:

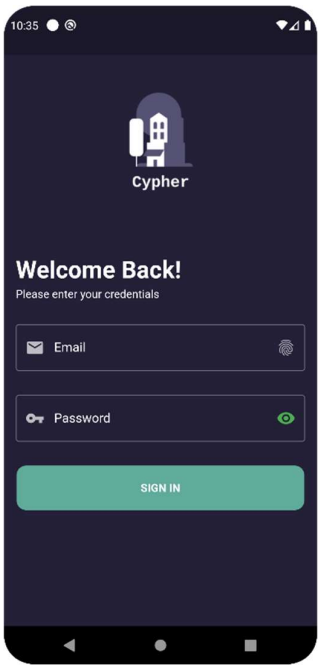

Figure 8 login page screenshot

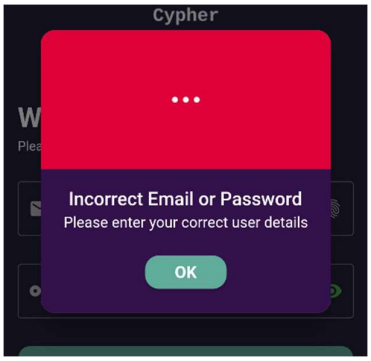

Figure 10 login page incorrect dialog screenshot

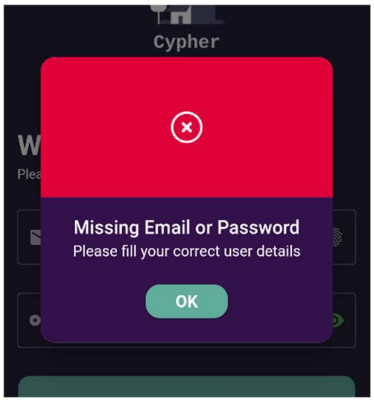

Figure 9 login page missing dialog screenshot

Our app features a login screen that utilizes Firebase authentication and supports logging in via Face ID or fingerprint recognition. Upon successful login, the app securely saves and encrypts the user's credentials for future use. If login info incorrect or missing error dialogs appears,

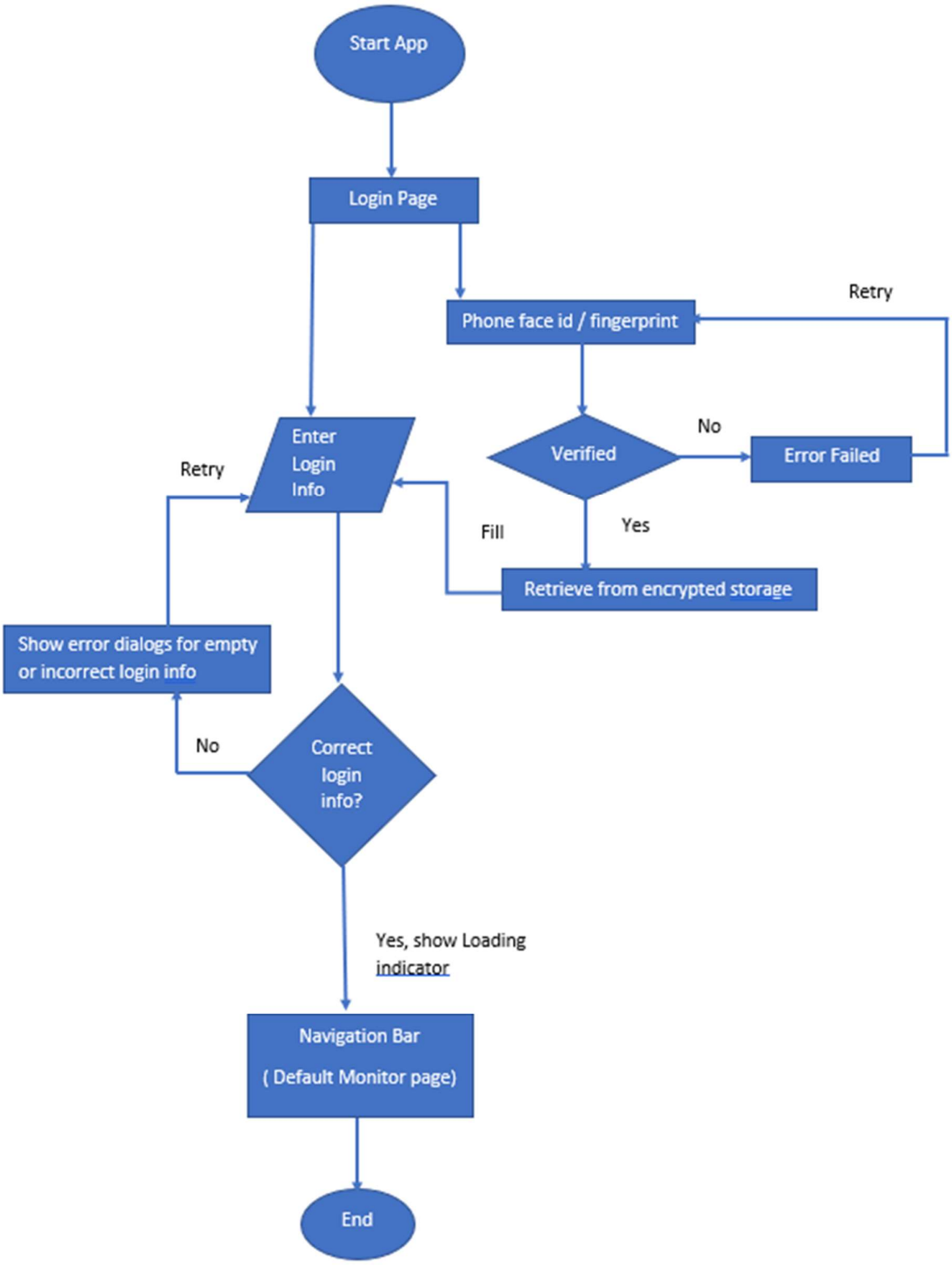

Figure 11 login page flow chart

Navigation bar :

The **navigation bar** offers access to the monitor page (default), profile page, history page, and settings page.

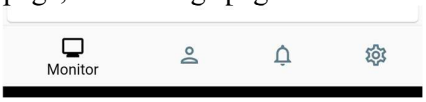

Figure 12 Navigation Bar

Monitor page:

The monitor page serves as a dashboard, displaying a personalized greeting based on the time of day and providing real-time updates on the status of the door lock and motion detection (using PIR and ultrasonic sensors).

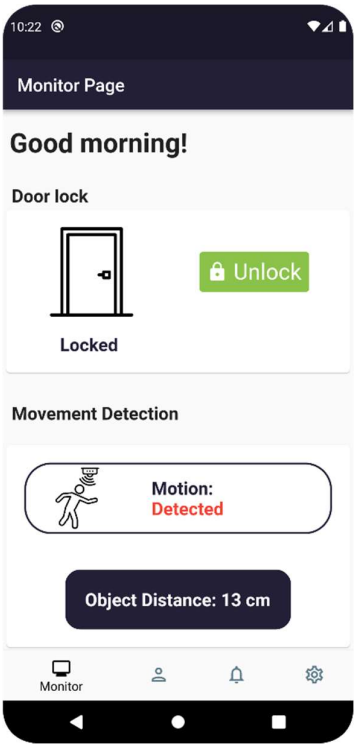

Figure 13 monitor page screenshot

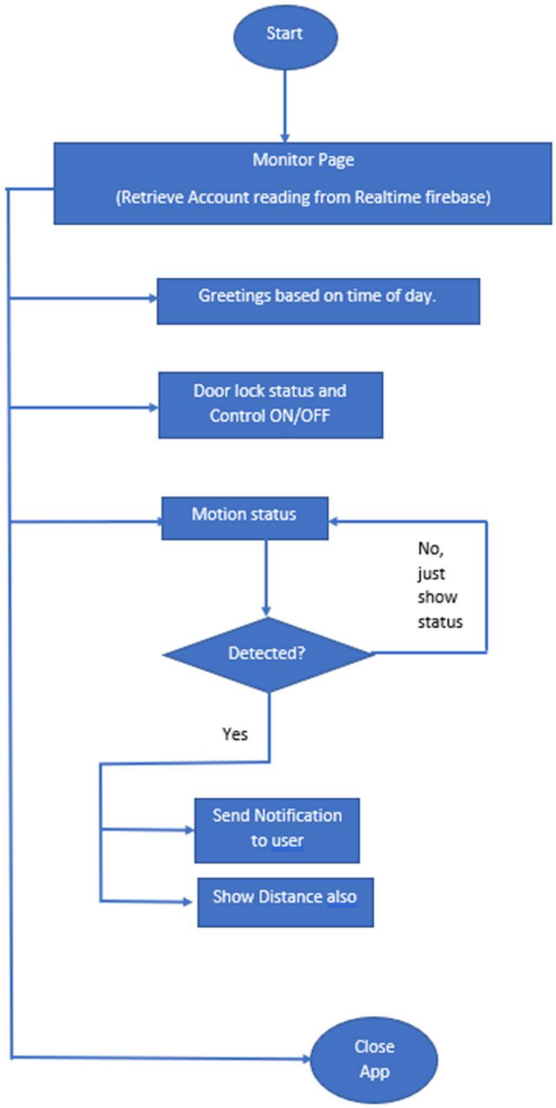

Figure 14 monitor page flow chart

Profile page:

The profile page allows for the creation and deletion of user profiles, and includes a fingerprint page where users can add (setup fingerprint) and delete their fingerprints.

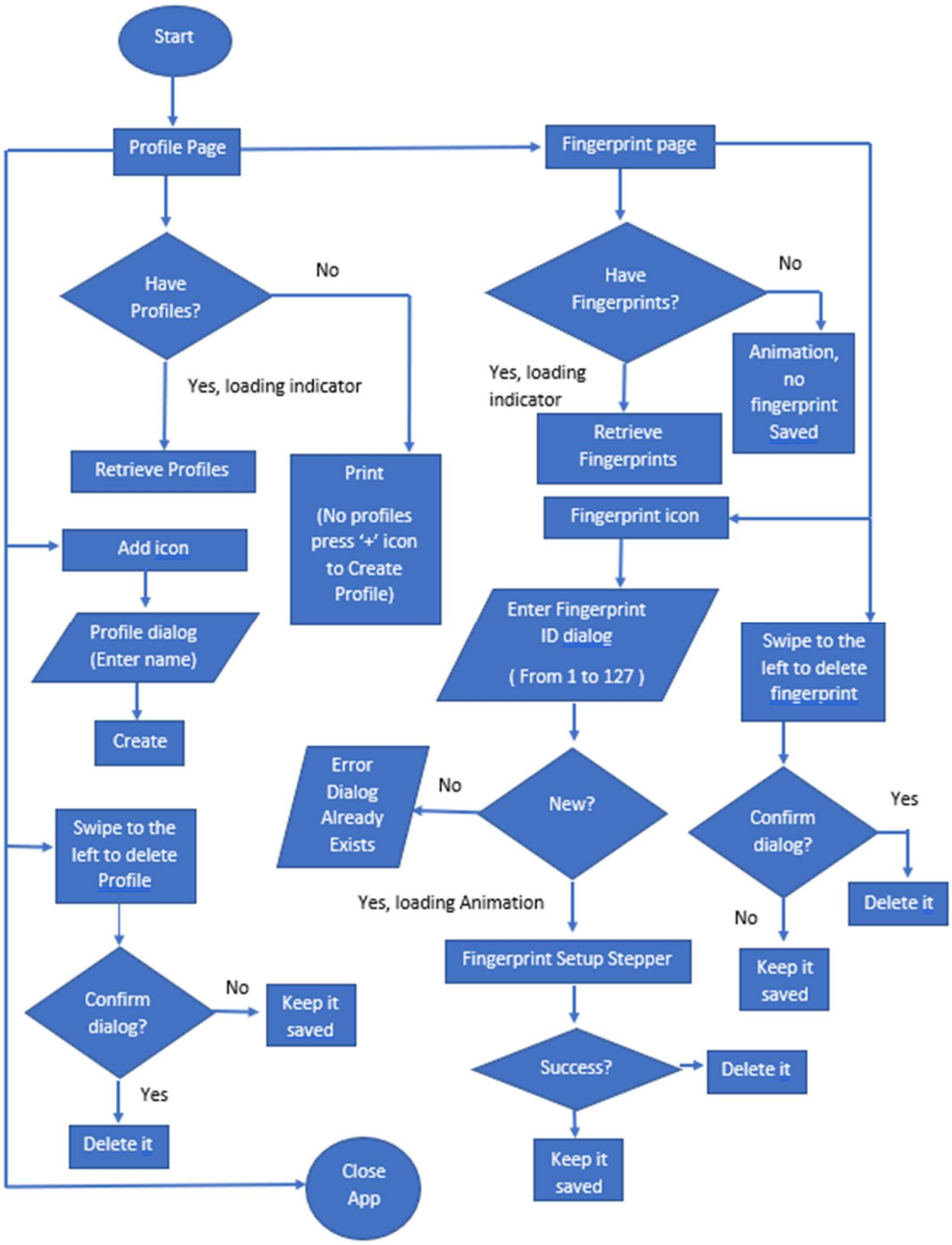

Figure 15 profile page flow chart

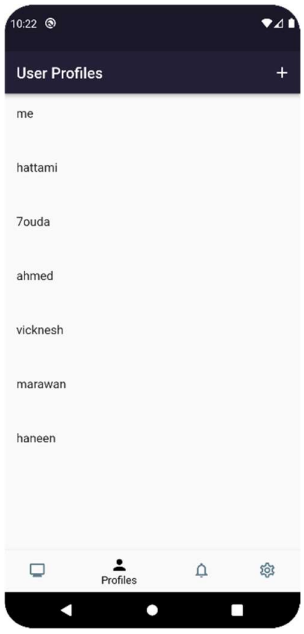

Figure 21 profile page screenshot

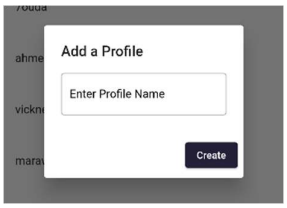

Figure 16 create profile dialog screenshot

Figure 17 fingerprints page empty screenshot

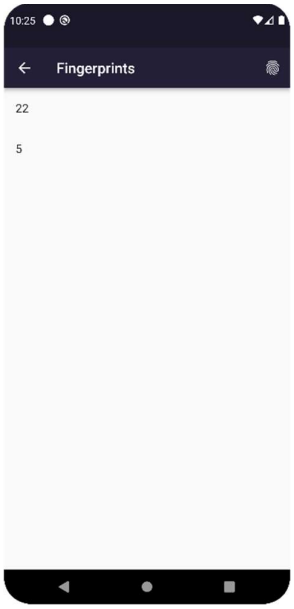

Figure 18 fingerprints page with ids screenshot

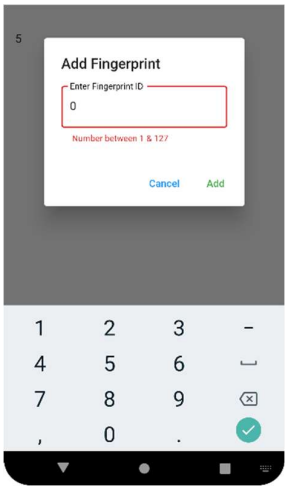

Figure 20 enter fingerprint id dialog with validation and numeric keyboard screenshot

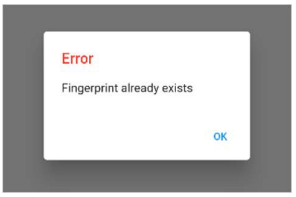

Figure 19 error dialog if fingerprints trying to add exists screenshot

Figure 24 loading animation for setup fingerprint screenshot

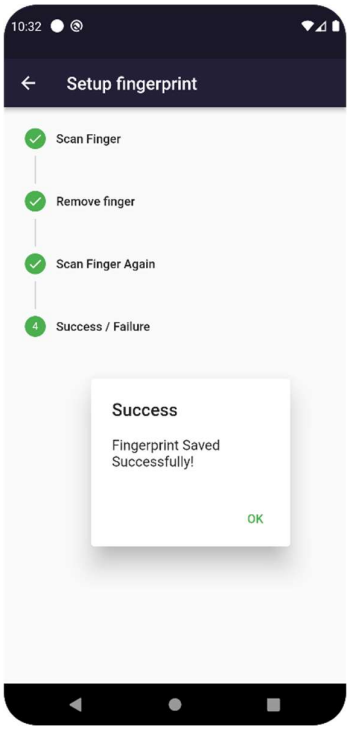

Figure 23 fingerprint stepper with success screenshot

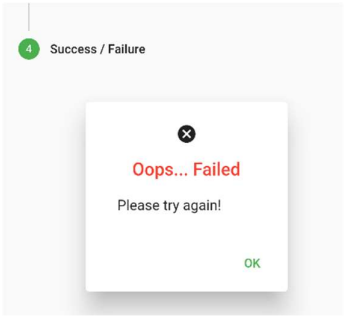

Figure 22 fingerprint stepper failed dialog screenshot

History page:

The history page displays a log of past entries into the home, recorded by the fingerprint sensor.

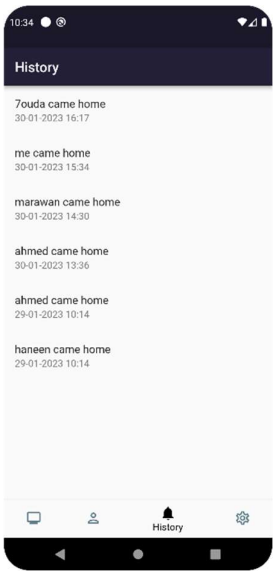

Figure 25 history page with data screenshot

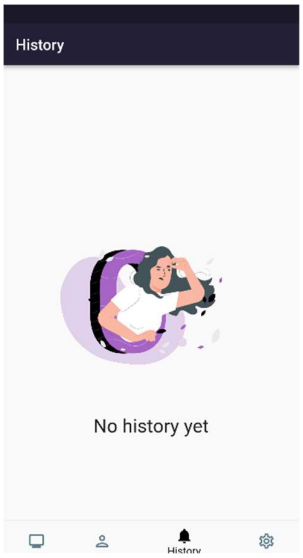

Figure 26 empty history page animation screenshot

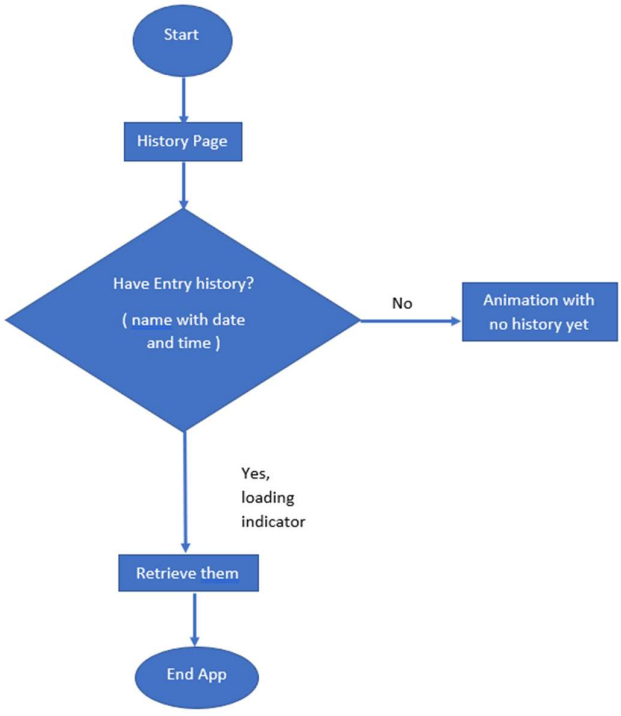

Figure 27 history page flow chart

Settings page:

The settings page includes options for managing account email, connecting to Wi-Fi, making emergency calls, and signing out of the app.

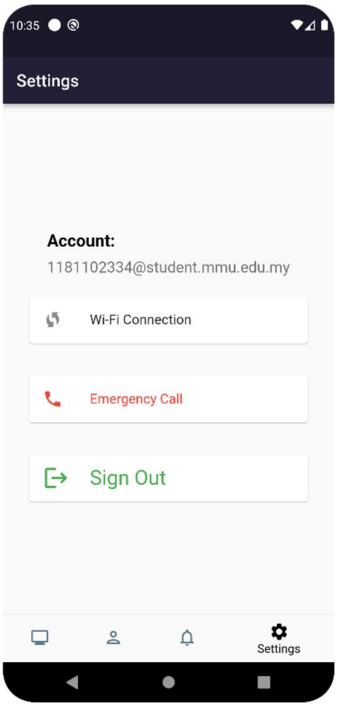

Figure 30 Settings page screenshot

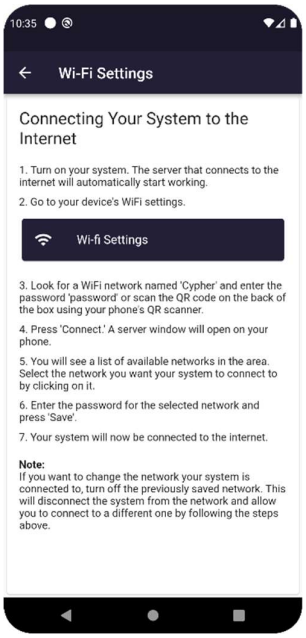

Figure 29 wifi settings page screenshot

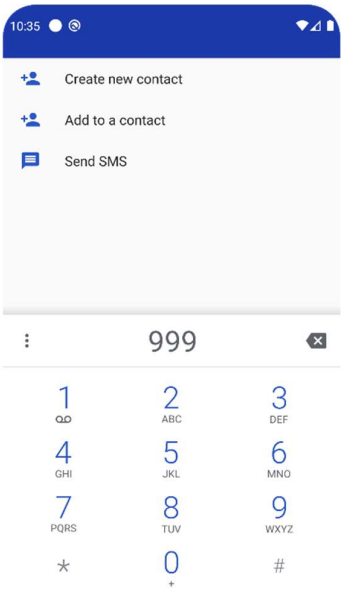

Figure 28 phone call with auto filled emergency number screenshot

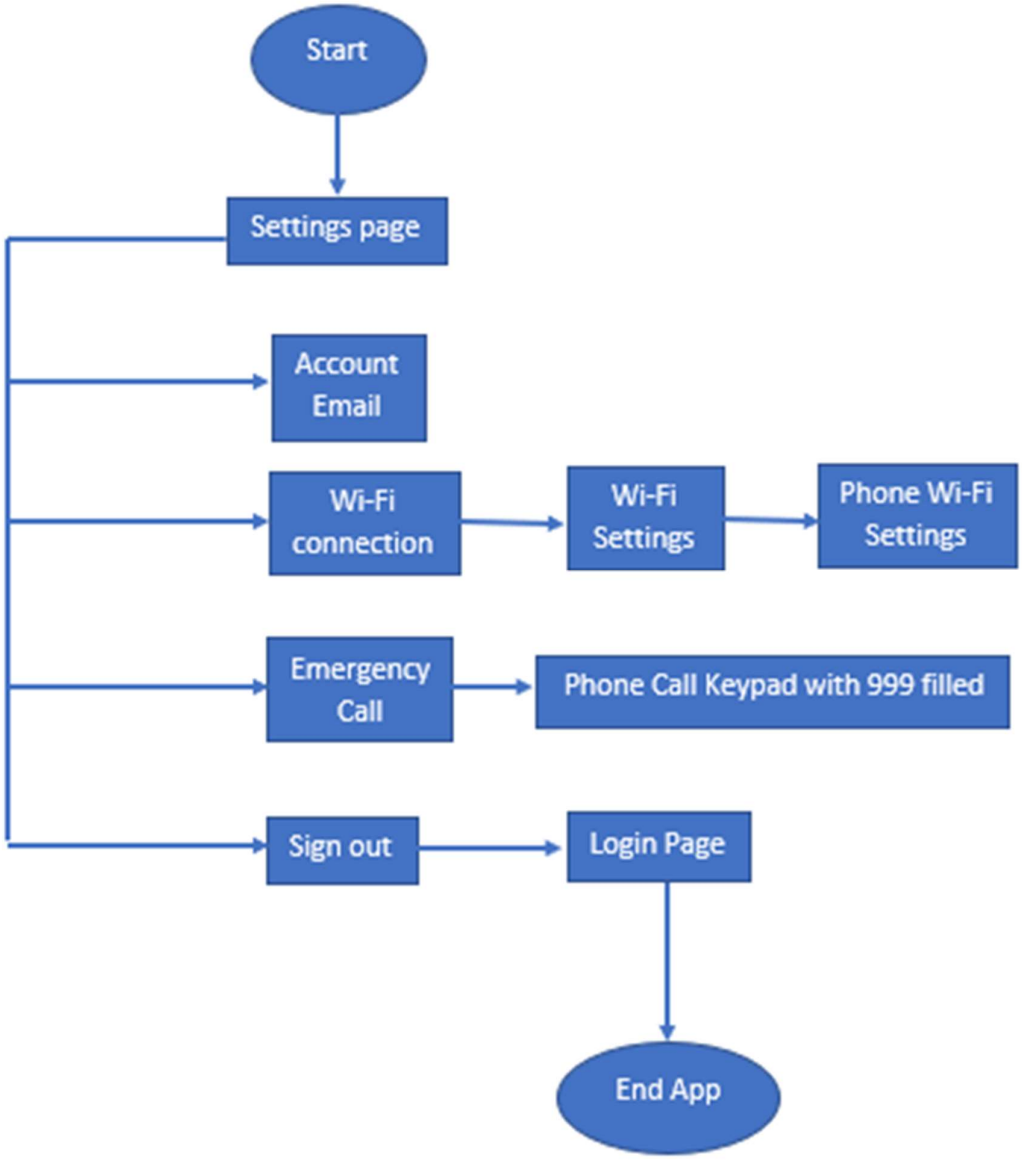

Figure 31 settings page flow chart

How to connect Project to Home Wi-Fi :

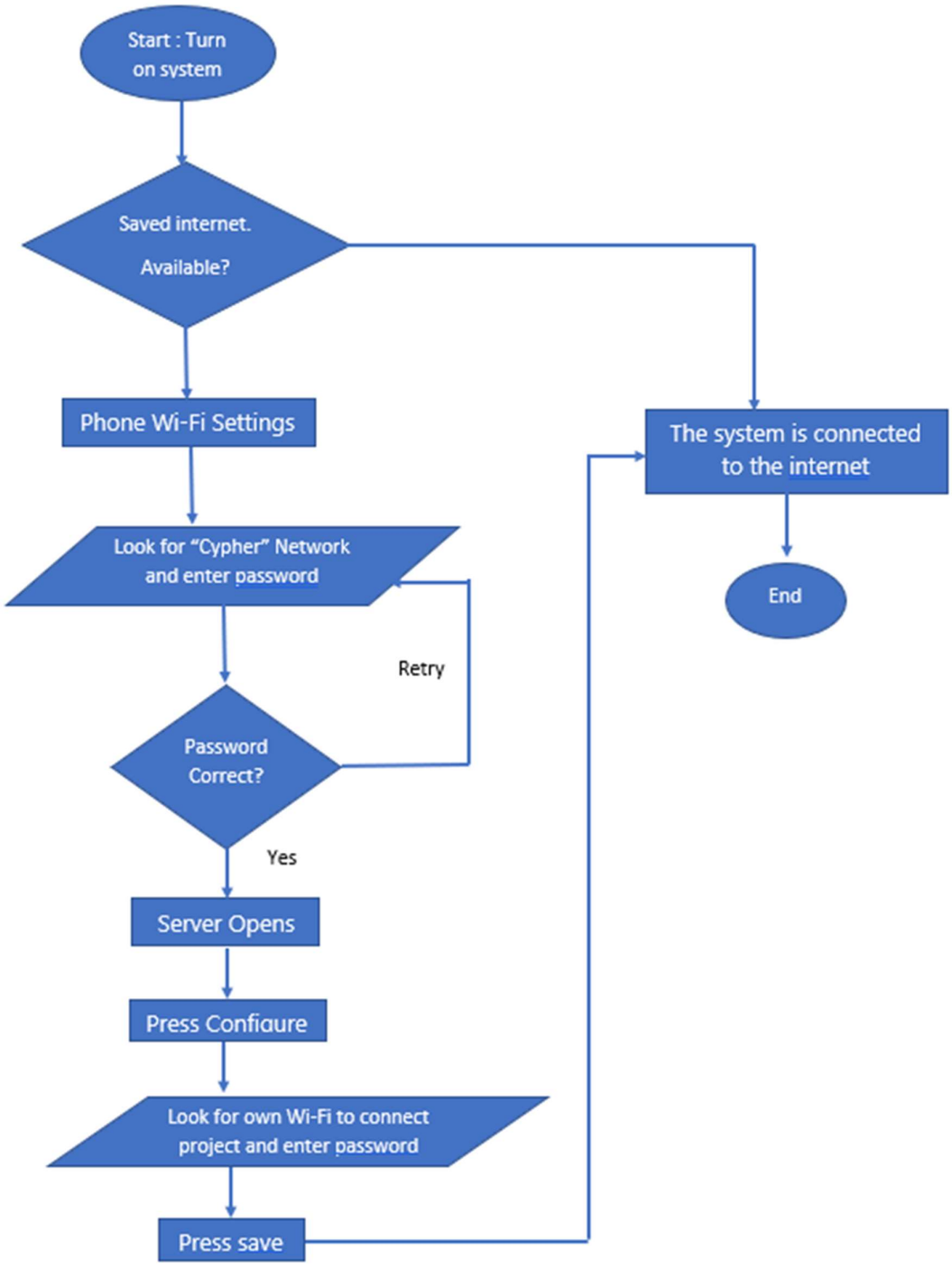

Figure 32 connect project to home wifi using server flow chart

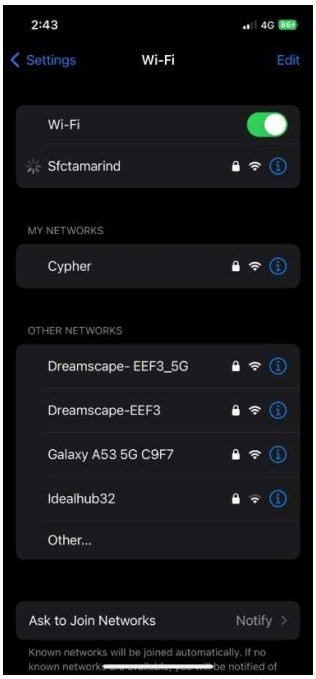

Figure 35 Cypher network in available netwok screenshot

Figure 33 server screenshot press configure

Figure 34 connect project to home network screenshot

```
23:37:08.326 -> *wm:AutoConnect
23:37:08.326 -> *wm:Connecting to SAVED AP: ahmed
23:37:08.830 -> *wm:connectTimeout not set, ESP waitForConnectResult...
23:37:14.834 -> *wm:AutoConnect: FAILED
23:37:14.834 -> *wm:StartAP with SSID:  Cypher
23:37:16.722 -> *wm:AP IP address: 192.168.4.1
23:37:16.766 -> *wm:Starting Web Portal
23:37:40.016 -> *wm:5 networks found
23:38:41.431 -> *wm:11 networks found
23:38:59.015 -> *wm:Connecting to NEW AP: ahmed
23:38:59.214 -> *wm:connectTimeout not set, ESP waitForConnectResult...
23:39:04.845 -> *wm:Connect to new AP [SUCCESS]
23:39:04.893 -> *wm:Got IP Address:
23:39:04.893 -> *wm:192.168.1.2
23:39:05.852 -> *wm:config portal exiting
23:39:05.852 -> connected...yeey :)
23:39:05.886 -> ahmed
23:39:05.886 -> Ahmed1937
```

Figure 36 arduino connection to server screenshot

Chapter 4: Presentation of Data and Discussion

4.1 Results and Discussion of Implementation (software)

The mobile application was tested for its performance and reliability. During the 4-hour test, the system functioned without any technical issues and had a quick response time for launching the app and sending notifications. The development process faced some challenges such as hardware compatibility issues, unexpected behavior of components, and compatibility with different platforms. To resolve these issues, use various debugging tools in android studio like flutter inspector, experimented with different devices, and searched online for solutions.

Some difficulties faced in the development process included programming logic, such as adding fingerprints to profiles and displaying history. To provide an easy-to-use interface, a delete function was added by swiping to the left. The setup process for the fingerprint was made user-friendly by using a stepper widget, which posed additional challenges in terms of updates. Overall, the results of the implementation demonstrate that the system is reliable and performs as intended. Any challenges faced during the development process were addressed and resolved to ensure a successful outcome.

Table 1 perfomance test

launch	4 seconds
notification	1 second
To update firebase	instance
Retrieve from firebase profiles and fingerprints or history.	1 second

And on customer feedback and capstone day it works with no technical issues at all when users tried it, all conditions were considered when developing the app since app tested test many times by shown them to friends and family to give comments on how to improve and to fix any technical issues they faced.

4.2 Analysis on the impact of the project (sustainability, the society and the environment)

Sustainability:

The smart door lock system presents a sustainable solution as it features a power-saving mechanism that conserves energy, reducing the reliance on electrical power and reducing energy costs for the user.

Furthermore, the use of IoT technology and the integration of multiple sensors result in a more efficient and reliable security system, reducing the need for frequent replacements and maintenance.

Impact on Society:

The system has a positive impact on society by providing a low-cost, user-friendly, and easy-to-use solution for enhancing home security.

The mobile application with a download QR code and instructions on the door makes setup simple and easy for the customer.

The real-time notifications and mobile application also allow the user to monitor their home from anywhere, providing greater convenience and flexibility.

The system's simple housing design is easy to install, meeting the needs of all customers.

Impact on the Environment:

The use of technology in the system reduces the environmental impact by reducing the need for physical keys, which often end up being discarded and contributing to environmental waste.

Additionally, the power-saving mechanism helps to reduce the overall energy consumption and carbon footprint of the system.

In conclusion, the IOT-based smart door lock system presents a sustainable, socially responsible, and environmentally friendly solution for enhancing home security, that is easy to use and install.

Chapter 5: Project Management

5.1 Project planning

Overall:

Activities	W 1	W 2	W 3	W 4	W 5	W 6	W 7	W 8	W 9	W 10	W 11	W 12	W 13	W 14	W 15	W 16
Defining problem statement and topic	■	■	■													
Literature Review		■	■													
Customer needs analysis survey		■	■													
List components			■													
Proposal presentation				■												
Purchase Components				■	■											
Research on the connection and model of the systems				■	■											
Build basic framework for mobile app				■	■											
Construct both IoT sensor and actuator systems				■	■											
Milestone (1) presentation						■										
Set-up Firebase						■	■									
Integrate the whole system using firebase server							■	■	■							
Test run the whole system								■	■							
Add enhancement									■	■						
Solve issues											■	■	■	■		
Customer feedback analysis survey														■		
Report writing														■	■	
Final presentation																■

Figure 37 Overall gantt chart

Individual:

Activities	W 1	W 2	W 3	W 4	W 5	W 6	W 7	W 8	W 9	W 10	W 11	W 12	W 13	W 14	W 15	W 16
Defining problem statement and topic	■	■	■													
Literature Review		■	■													
Customer needs analysis survey		■	■													
Download android studio and setup			■													
Proposal presentation				■												
Build a prototype for mobile app and evaluate efficiency				■	■											
Milestone (1) presentation				■	■											
Set-up Firebase and link to app						■										
Create login authentication						■	■									
Navigation bar						■	■									
Monitor page							■	■	■							
Settings page							■	■	■							
History page							■	■	■							
Login with finger or face id to app instead of entering email and password							■	■	■							
Test run the whole system								■	■							
Add enhancement (setup fingerprint using application)									■	■	■					
Solve issues											■	■	■	■		
Customer feedback analysis														■		
Report writing														■	■	
Final presentation																■

Figure 38 individual gantt chart

5.2 Budget planning and cost analysis

Table 2 components list

No.	Component name	Per unit price (RM)	no. units	Total price (RM)
1	Fingerprint sensor	54.50	1	54.50
2	NodeMCU ESP8266	14.90	1	14.90
3	Transistor IRFZ44N	5	1	5
4	12VDC Solenoid door Lock	17	1	17
5	Jumper wires M-M	2	1	2
6	Jumper wires F-M	2	1	2
7	HC-SR501 PIR Sensor	3.3	1	3.3
8	5VDC HC-SR04 ultrasonic sensor	4.5	1	4.5
10	Voltage regulator +5V	1	1	1
11	Piezo Buzzer with Wires	1	1	1
12	10k ohm resistor	0.05	1	0.05
13	Voltage regulator +3.3V	2	1	2
14	Housing	10	3	30
15	10uf capacitor	0.4	4	1.6
Total price:				138.85

Chapter 6: Conclusions and Recommendations

6.1 Summary on the work done.

The IOT based smart door lock system project has successfully been completed and the results of the study have shown that it can be used effectively for home security. The use of a PIR sensor and ultrasonic sensor for motion detection, a fingerprint sensor for unlocking the door, and a mobile application for remote monitoring and control have been successful in providing a secure and efficient security system. The use of Firebase IoT platform has made it possible for the system to be monitored in real-time and the ability to receive alerts via phone and buzzer have made it possible for homeowners to be informed of any potential security breaches.

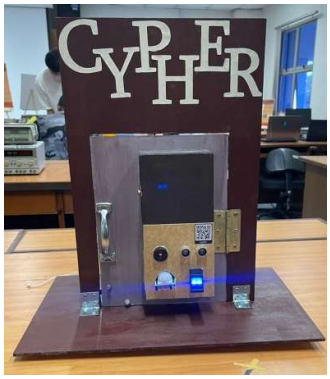

Figure 40 Front view of door lock system project

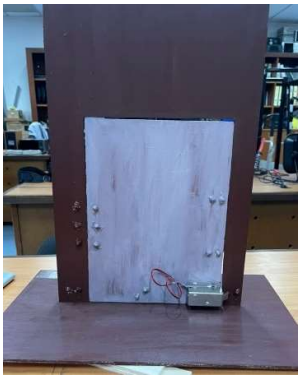

Figure 39 Back view

Figure 43 Inside view

Figure 41 qr code for cypher app.

cypher

Figure 42 cypher internet qr

6.2 Recommendation for future improvement

While the system is highly efficient, there is always room for improvement. In the future, it can be improved by adding more advanced sensors to detect different types of security breaches and by incorporating artificial intelligence algorithms to make the system even smarter. Another potential improvement could be the integration of voice control so that the system can be activated and deactivated through voice commands. Also, Can add keypad lock and implement inside and outside the house. Finally, in software making the system automatically assign the fingerprint id not the user could make this system even more convenient and user-friendly.

References:

https://pub.dev/packages/google_nav_bar

https://pub.dev/packages/line_icons

https://pub.dev/packages/firebase_auth

https://pub.dev/packages/flutter_login

https://pub.dev/packages/quickalert

https://pub.dev/packages/firebase_core

https://pub.dev/packages/firebase_database

https://pub.dev/packages/local_auth

https://pub.dev/packages/flutter_secure_storage

https://pub.dev/packages/open_settings

https://pub.dev/packages/awesome_notifications

https://pub.dev/packages/intl

https://pub.dev/packages/url_launcher

https://pub.dev/packages/flutter_spinkit

https://pub.dev/packages/lottie

https://pub.dev/packages/flutter_launcher_icons'

https://github.com/tzapu/WiFiManager

https://firebase.flutter.dev/docs/overview/

https://docs.flutter.dev/

https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Library

Appendices (optional)

https://github.com/Marawan2002/Cypherflutter.git

https://forms.gle/wieHqJBtQYT6kZsX9

https://forms.gle/3gWwXdPgkXtXX3mh9

https://console.firebase.google.com/project/cypherflutter-4e3d3/database/cypherflutter-4e3d3-default-rtdb/data?pli=1

App code:

Main.dart:

```
// ignore_for_file: prefer_const_constructors
import 'package:cypherflutter/widget/mp.dart';
import 'package:firebase_database/firebase_database.dart';
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/services.dart';
import 'navpages/noti.dart';

Future main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  await NotificationController.initializeLocalNotifications();
  SystemChrome.setPreferredOrientations(
    [DeviceOrientation.portraitUp, DeviceOrientation.portraitDown],
  );
  runApp(MyApp());
}

final navigatorKey = GlobalKey<NavigatorState>();

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

  @override
  State<MyApp> createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  DatabaseReference databaseRef = FirebaseDatabase.instance.ref();

  @override
  void initState() {
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      //navigatorKey: navigatorKey,
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
```

```
        primarySwatch: Colors.blue,
      ),
      home: mp(),
    );
  }
}
```

mp.dart:

```dart
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter_spinkit/flutter_spinkit.dart';
import '../navpages/navigationbar.dart';
import 'constants.dart';
import 'login_widget.dart';

class mp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: StreamBuilder<User?>(
        stream: FirebaseAuth.instance.authStateChanges(),
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return Center(
                child: SpinKitPouringHourGlassRefined(
                  color: appcolortheme,
                ));
          } else if (snapshot.hasError) {
            print('something wrong');
            return Center(child: Text('Something went wrong!'));
          } else if (snapshot.hasData) {
            return navigationbar();
          } else {
            print('login widget');
            return LoginWidget();
          }
        },
      ),
    );
  }
}
```

login_widget.dart

```
import 'dart:async';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:flutter_spinkit/flutter_spinkit.dart';
import 'package:quickalert/models/quickalert_type.dart';
import 'package:quickalert/widgets/quickalert_dialog.dart';
import 'constants.dart';
import 'package:local_auth/local_auth.dart';
import 'package:flutter_secure_storage/flutter_secure_storage.dart';

class LoginWidget extends StatefulWidget {
  @override
  State<LoginWidget> createState() => _LoginWidgetState();
}

class _LoginWidgetState extends State<LoginWidget> {
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();
  final LocalAuthentication auth = LocalAuthentication();
  final _storage = const FlutterSecureStorage();
  _SupportState _supportState = _SupportState.unknown;
  String _authorized = 'Not Authorized';
  bool _isAuthenticating = false;
  bool _isSaved = false;
  late String _email;
  late String _password;
  bool _obscureText = true;

  @override
  void initState() {
    super.initState();
    _checkSavedCredentials();
    auth.isDeviceSupported().then(
          (bool isSupported) => setState(() => _supportState =
isSupported
              ? _SupportState.supported
              : _SupportState.unsupported),
        );
  }

  Future<void> _saveCredentials() async {
    await _storage.write(key: 'email', value:
_emailController.text);
    await _storage.write(key: 'password', value:
_passwordController.text);
  }

  Future<void> _checkSavedCredentials() async {
    final email = await _storage.read(key: 'email');
    final password = await _storage.read(key: 'password');
    if (email != null && password != null) {
      _isSaved = true;
      _email = email;
      _password = password;
    }
  }
```

```
  Future<void> _authenticate() async {
    bool authenticated = false;
    try {
      setState(() {
        _isAuthenticating = true;
        _authorized = 'Authenticating';
      });
      authenticated = await auth.authenticate(
        localizedReason: 'Touch the fingerprint sensor',
        options: const AuthenticationOptions(
          stickyAuth: true,
        ),
      );
      setState(() {
        _isAuthenticating = false;
      });
    } on PlatformException catch (e) {
      print(e);
      setState(() {
        _isAuthenticating = false;
        _authorized = 'Error - ${e.message}';
      });
      return;
    }
    if (!mounted) {
      return;
    }
    if (_isSaved && authenticated) {
      _emailController.text = _email;
      _passwordController.text = _password;
    }
    setState(
        () => _authorized = authenticated ? 'Authorized' : 'Not
Authorized');
  }

  Future signIn() async {
    showDialog(
        context: context,
        barrierDismissible: false,
        builder: (context) {
          return const Center(
            child: SpinKitFadingFour(
              color: appcolortheme,
            ),
          );
        });
    try {
      await FirebaseAuth.instance.signInWithEmailAndPassword(
        email: _emailController.text.trim(),
        password: _passwordController.text.trim(),
      );
      await _saveCredentials();
      Navigator.pop(context);
    } on FirebaseAuthException catch (e) {
      print(e);
      if (e.code == 'user-not-found' ||
          e.code == 'wrong-password' ||
```

```
          e.code == 'invalid-email') {
        Navigator.pop(context);
        return QuickAlert.show(
          context: context,
          type: QuickAlertType.error,
          title: 'Incorrect Email or Password',
          text: 'Please enter your correct user details',
          backgroundColor: kInactiveCardColor,
          titleColor: Colors.white,
          textColor: Colors.white,
          confirmBtnColor: kBottomContainerColor,
          confirmBtnText: 'OK',
        );
      } else if (e.code == 'unknown') {
        Navigator.pop(context);
        return QuickAlert.show(
          context: context,
          type: QuickAlertType.error,
          title: 'Missing Email or Password',
          text: 'Please fill your correct user details',
          backgroundColor: kInactiveCardColor,
          titleColor: Colors.white,
          textColor: Colors.white,
          confirmBtnColor: kBottomContainerColor,
          confirmBtnText: 'OK',
        );
      }
    }
  }

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData.dark().copyWith(
        appBarTheme: AppBarTheme(color: Color(0xFF242038)),
        scaffoldBackgroundColor: appcolortheme,
        textTheme: TextTheme(bodyMedium: TextStyle(color:
Colors.white)),
      ),
      home: Scaffold(
        resizeToAvoidBottomInset: false,
        body: Padding(
          padding: const EdgeInsets.all(15.0),
          child: SingleChildScrollView(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.start,
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Center(
                  child: Container(
                    child: Image.asset(
                      'assets/logo_foreground.png',
                      height: 300,
                    ),
                  ),
                ),
                Text(
                  'Welcome Back!',
```

```
                  style: kResultTextStyle,
                ),
                SizedBox(
                  height: 5,
                ),
                Text(
                  'Please enter your credentials',
                ),
                SizedBox(
                  height: 30,
                ),
                Container(
                  child: TextField(
                    controller: _emailController,
                    keyboardType: TextInputType.emailAddress,
                    decoration:  InputDecoration(
                      focusedBorder: OutlineInputBorder(
                        borderSide: BorderSide(color: Colors.blue),
                      ),
                      border: OutlineInputBorder(),
                      prefixIcon: Icon(Icons.email),
                      labelText: 'Email',
                      suffixIcon: GestureDetector(
                          onTap: () => _authenticate(),
                      child: Icon(Icons.fingerprint,),),
                      labelStyle: kLabelTextStyle,
                    ),
                  ),
                ),
                SizedBox(
                  height: 30,
                ),
                Container(
                  child: TextField(
                    obscureText: _obscureText,
                    controller: _passwordController,
                    decoration: InputDecoration(
                      focusedBorder: OutlineInputBorder(
                          borderSide: BorderSide(color:
Colors.blue)),
                      border: OutlineInputBorder(),
                      prefixIcon: Icon(Icons.vpn_key),
                      suffixIcon: GestureDetector(
                        onTap: () {
                          setState(() {
                            _obscureText = !_obscureText;
                          });
                        },
                        child: Icon(_obscureText
                            ? Icons.visibility
                            : Icons.visibility_off),
                      ),
                      suffixIconColor: _obscureText ? Colors.green :
Colors.red,
                      labelText: 'Password',
                      labelStyle: kLabelTextStyle,
                    ),
                  ),
                ),
```

```
                SizedBox(
                  height: 30,
                ),
                Padding(
                  padding: const EdgeInsets.symmetric(horizontal:
1.0),
                  child: GestureDetector(
                    onTap: signIn,
                    child: Container(
                      padding: EdgeInsets.all(20),
                      child: Center(
                        child: Text(
                          'SIGN IN',
                          textAlign: TextAlign.center,
                          style: TextStyle(fontWeight:
FontWeight.w700),
                        ),
                      ),
                      decoration: BoxDecoration(
                        color: kBottomContainerColor,
                        borderRadius: BorderRadius.circular(12.0),
                      ),
                    ),
                  ),
                ),
              ],
            ),
          ),
        ),
      ),
    );
  }
}

enum _SupportState {
  unknown,
  supported,
  unsupported,
}
```

constants.dart:

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


const kActiveCardColor = Color(0xFF592E83);
const kInactiveCardColor = Color(0xFF33114A);
const kBottomContainerColor = Color(0xFF60AB9A);
const kCardTextStyle = TextStyle(
  fontSize: 18.0,
  color: Color(0xFFbdb0e3),
);

const kLargeButtonTextStyle = TextStyle(
  fontSize: 20.0,
  fontWeight: FontWeight.bold,
);
const kResultTextStyle = TextStyle(
  fontSize: 30.0,
  fontWeight: FontWeight.w600,
);

const kLabelTextStyle = TextStyle(
  fontSize: 16.0,
  color: Color(0xFFFFFFFF),
);

const appcolortheme =  Color(0xFF242038);
```

noti.dart:

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

class NotificationController {
  static Future<void> initializeLocalNotifications() async {
    await AwesomeNotifications().initialize(
        null,
        [
          NotificationChannel(
              channelKey: 'alerts',
              channelName: 'Alerts',
              channelDescription: 'Notification tests as alerts',
              playSound: true,
              onlyAlertOnce: true,
              channelShowBadge: true,
              groupAlertBehavior: GroupAlertBehavior.Children,
              importance: NotificationImportance.High,
              defaultPrivacy: NotificationPrivacy.Private,
              defaultColor: Colors.deepPurple,
              ledColor: Colors.deepPurple)
        ],
        debug: true);
  }

  static Future<void> createNewNotification() async {
    bool isAllowed = await
AwesomeNotifications().isNotificationAllowed();
    if (!isAllowed) return;

    await AwesomeNotifications().createNotification(
      content: NotificationContent(
          id: -1,
          channelKey: 'alerts',
          title: 'Motion detected',
          body: 'Motion has been detected at front door',
          notificationLayout: NotificationLayout.BigText,
          criticalAlert: true,
          displayOnForeground: true,
      displayOnBackground: true),
      actionButtons: [
        NotificationActionButton(
            key: 'DISMISS',
            label: 'Dismiss',
            actionType: ActionType.DismissAction,
            isDangerousOption: true),

      ],
    );
  }
}
```

navigationbar.dart:

```
import 'package:cypherflutter/navpages/profile_page.dart';
import 'package:cypherflutter/navpages/history_page.dart';
import 'package:cypherflutter/navpages/monitor_page.dart';
import 'package:cypherflutter/navpages/settings_page.dart';
import 'package:flutter/material.dart';

late final Widget activeIcon;
late final Widget icon;

class navigationbar extends StatefulWidget {
  const navigationbar({Key? key}) : super(key: key);

  @override
  State<navigationbar> createState() => _navigationbarState();
}

class _navigationbarState extends State<navigationbar> {
  List pages = [
    monitorpage(),
    ProfileListPage(),
    HistoryPage(),
    settingspage(),
  ];
  int currentIndex = 0;

  void onTap(int index) {
    setState(() {
      currentIndex = index;
    });
  }

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

      backgroundColor: Colors.white,
      body: pages[currentIndex],
      bottomNavigationBar: BottomNavigationBar(
        showSelectedLabels: true,
        showUnselectedLabels: false,
        type: BottomNavigationBarType.shifting,
        onTap: onTap,
        currentIndex: currentIndex,
        selectedItemColor: Colors.black,
        unselectedItemColor: Colors.blueGrey,
        items: [
          BottomNavigationBarItem(
              icon: Icon(Icons.monitor_outlined),
              label: 'Monitor',
              activeIcon: Icon(Icons.monitor)),
          BottomNavigationBarItem(
              icon: Icon(Icons.person_outlined),
              label: 'Profiles',
              activeIcon: Icon(Icons.person)),
          BottomNavigationBarItem(
              icon: Icon(Icons.notifications_outlined),
              label: 'History',
```

```
              activeIcon: Icon(Icons.notifications)),
          BottomNavigationBarItem(
              icon: Icon(Icons.settings_outlined),
              label: 'Settings',
              activeIcon: Icon(Icons.settings)),
        ],
      ),
    );
  }
}
```

monitor_page.dart:

```dart
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:firebase_database/firebase_database.dart';
import 'package:flutter_spinkit/flutter_spinkit.dart';
import '../widget/constants.dart';
import 'noti.dart';

class monitorpage extends StatefulWidget {
  const monitorpage({Key? key}) : super(key: key);

  @override
  State<monitorpage> createState() => _monitorpageState();
}

extension BoolParsing on String {
  bool parseBool() {
    return this.toLowerCase() == 'true';
  }
}

class _monitorpageState extends State<monitorpage> {
  FirebaseDatabase database = FirebaseDatabase.instance;
  final user = FirebaseAuth.instance.currentUser!;
  DatabaseReference databaseRef = FirebaseDatabase.instance.ref();

  int? _distance;
  bool? _doorLocked;
  bool? _motion;

  void _unlockDoor() async {
    // update the status of door in firebase
    _doorLocked = false;
    await databaseRef.child(user.uid).update({
      'doorLocked': _doorLocked,
    });
  }

  void _lockDoor() async {
    // update the status of door in firebase
    _doorLocked = true;
    await databaseRef.child(user.uid).update({
      'doorLocked': _doorLocked,
    });
  }

  void _activateListeners() {
    databaseRef
        .child(user.uid.toString())
        .child('distance')
        .onValue
        .listen((event) {
      setState(() {
        _distance = int.parse(event.snapshot.value.toString());
      });
    });

    databaseRef
```

```
        .child(user.uid.toString())
        .child('doorLocked')
        .onValue
        .listen((event) {
      setState(() {
        _doorLocked = event.snapshot.value.toString().parseBool();
      });
    });

    databaseRef
        .child(user.uid.toString())
        .child('motion')
        .onValue
        .listen((event) {
      setState(() {
        _motion = event.snapshot.value.toString().parseBool();
      });
      if (_motion == true) {
// show notification when motion is detected
        NotificationController.createNewNotification();
      }
    });
  }

  @override
  void initState() {
    super.initState();
    _activateListeners();
  }

  @override
  Widget build(BuildContext context) {
    final now = DateTime.now();
    String greeting = '';
    if (now.hour < 12) {
      greeting = ' Good morning!';
    } else if (now.hour < 17) {
      greeting = ' Good afternoon!';
    } else {
      greeting = ' Good evening!';
    }
    return Scaffold(
      appBar: AppBar(
        title: Text('Monitor Page'),
        backgroundColor: appcolortheme,
      ),
      body: SingleChildScrollView(
        child: SafeArea(
          child: _distance == null || _doorLocked == null
              ? Center(
                  child: SpinKitFadingFour(
                  color: appcolortheme,
                ))
              : Column(
                  mainAxisAlignment: MainAxisAlignment.center,
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    SizedBox(
                      height: 20,
```

```
                    ),
                    Text(
                      greeting,
                      style: TextStyle(
                        fontSize: 30,
                        fontWeight: FontWeight.bold,
                      ),
                    ),
                    SizedBox(
                      height: 30,
                    ),
                    Text(
                      '  Door lock',
                      style: TextStyle(
                        fontSize: 20,
                        fontWeight: FontWeight.bold,
                      ),
                    ),
                    Card(
                      child: Container(
                        padding: EdgeInsets.all(20),
                        child: Column(
                          children: [
                            Row(
                              mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
                              children: [
                                Image.asset(
                                  _doorLocked == true
                                      ? 'assets/doorlocked.png'
                                      : 'assets/door.png',
                                  height: 100,
                                ),
                                SizedBox(width: 20),
                                TextButton(
                                  child: RichText(
                                    text: TextSpan(
                                      children: [
                                        WidgetSpan(
                                          child: _doorLocked ??
false
                                              ? Icon(
                                                  Icons.lock,
                                                  color:
Colors.white,
                                                )
                                              : Icon(
                                                  Icons.lock_open,
                                                  color:
Colors.white,
                                                ),
                                        ),
                                        TextSpan(
                                          text: _doorLocked ?? false
                                              ? ' Unlock'
                                              : ' Lock',
                                          style: TextStyle(
                                            color: Colors.white,
                                            fontSize: 25,
```

```
                                        ),
                                    ),
                                ],
                            ),
                        ),
                        style: TextButton.styleFrom(
                          foregroundColor: Colors.white,
                          backgroundColor: _doorLocked ??
false
                              ? Colors.lightGreen
                              : Colors.red,
                        ),
                        onPressed: _doorLocked ?? false
                            ? _unlockDoor
                            : _lockDoor,
                      ),
                    ],
                  ),
                  SizedBox(height: 20),
                  Row(
                    children: [
                      Text(
                        (_doorLocked == true
                            ? '        Locked'
                            : '        Unlocked'),
                        style: TextStyle(
                          fontSize: 20,
                          color: appcolortheme,
                          fontWeight: FontWeight.bold,
                        ),
                      ),
                      SizedBox(width: 100),
                    ],
                  ),
                ],
              ),
            ),
          ),
          SizedBox(
            height: 30,
          ),
          Text(
            '  Movement Detection',
            style: TextStyle(
              fontSize: 20,
              fontWeight: FontWeight.bold,
            ),
          ),
          SizedBox(
            height: 20,
          ),
          Card(
              child: Column(children: [
            Container(
              width: 390,
              height: 120,
              child: Padding(
                padding: EdgeInsets.all(20),
                child: Container(
```

```
                            decoration: BoxDecoration(
                              border: Border.all(
                                color: appcolortheme,
                                width: 2,
                              ),
                              borderRadius:
BorderRadius.circular(30),
                            ),
                            child: Row(
                              mainAxisAlignment:
MainAxisAlignment.spaceBetween,
                              children: <Widget>[
                                Container(
                                  margin: EdgeInsets.only(left: 20),
                                  width: 75,
                                  height: 75,
                                  child: Image.asset(
                                    'assets/motion.png',
                                    width: 100,
                                    height: 100,
                                    fit: BoxFit.cover,
                                  ),
                                ),
                                Container(
                                  width: 200,
                                  height: 150,
                                  child: Column(
                                    mainAxisAlignment:
MainAxisAlignment.center,
                                    crossAxisAlignment:
                                        CrossAxisAlignment.start,
                                    children: <Widget>[
                                      Text(
                                        'Motion:',
                                        style: TextStyle(
                                          fontSize: 20,
                                          color: appcolortheme,
                                          fontWeight:
FontWeight.bold,
                                        ),
                                      ),
                                      Text(
                                        _motion != null && _motion!
                                            ? 'Detected'
                                            : 'Not Detected',
                                        style: TextStyle(
                                          fontSize: 20,
                                          color: _motion != null &&
_motion!
                                              ? Colors.red
                                              : Colors.green,
                                          fontWeight:
FontWeight.bold,
                                        ),
                                      ),
                                    ],
                                  ),
                                ),
                              ],
```

```
                            ),
                          ),
                        ),
                      ),
                      Padding(
                        padding: const EdgeInsets.all(20.0),
                        child: Visibility(
                          visible: _motion ?? false,
                          child: Container(
                            padding: EdgeInsets.all(20),
                            decoration: BoxDecoration(
                              color: appcolortheme,
                              borderRadius:
BorderRadius.circular(17),
                              border:
                                  Border.all(width: 2, color:
appcolortheme),
                            ),
                            child: Text(
                              'Object Distance: ' +
                                  _distance.toString() +
                                  ' cm',
                              style: TextStyle(
                                fontSize: 20,
                                color: Colors.white,
                                fontWeight: FontWeight.bold,
                              ),
                            ),
                          ),
                        ),
                      )
                    ]))
                  ],
                ),
              ),
            ),
          );
        }
      }
```

profile_page.dart:

```
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_database/firebase_database.dart';
import 'package:flutter_spinkit/flutter_spinkit.dart';

import '../widget/constants.dart';
import 'FingerprintPage.dart';

class ProfileListPage extends StatefulWidget {
  @override
  _ProfileListPageState createState() => _ProfileListPageState();
}

String selectedProfileKey = '';

class _ProfileListPageState extends State<ProfileListPage> {
  FirebaseDatabase database = FirebaseDatabase.instance;
  final user = FirebaseAuth.instance.currentUser!;
  DatabaseReference databaseRef = FirebaseDatabase.instance.ref();

  String name = '';
  int? fingerprint;
  String profileKey = '';

  void _addProfile(String name) async {
    // Generate a push ID for the new profile
    DatabaseReference profileRef =
        databaseRef.child(user.uid).child('profiles').push();
    // Assign the push ID to the profileKey variable
    profileKey = profileRef.key!;
    // Use the generated push ID as the key for the new profile
    profileRef.set({'name': name});
  }

  void _deleteProfile(String profileKey) async {
    DatabaseReference profileRef =

databaseRef.child(user.uid).child('profiles').child(profileKey);
    profileRef.remove();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('User Profiles'),
        backgroundColor: appcolortheme,
        actions: [
          IconButton(
            icon: Icon(Icons.add),
            onPressed: () {
              showDialog(
                context: context,
                builder: (BuildContext context) {
                  return AlertDialog(
```

```
                    title: Text('Add a Profile'),
                    content: TextField(
                      onChanged: (value) {
                        // Set the value of the entered name to a
local variable
                        name = value;
                      },
                      decoration: InputDecoration(
                        focusedBorder: OutlineInputBorder(
                          borderSide: BorderSide(color:
Colors.lightGreen),
                        ),
                        border: OutlineInputBorder(),
                        labelText: 'Enter Profile Name',
                        labelStyle: TextStyle(color: Colors.black),
                      ),
                    ),
                    actions: [
                      ElevatedButton(
                        child: Text('Create'),
                        onPressed: () {
                          // Pass the value of the entered name to
the _addProfile method
                          _addProfile(name);
                          Navigator.of(context).pop();
                        },
                        style: ButtonStyle(
                            backgroundColor:

MaterialStateProperty.all(appcolortheme)),
                      ),
                    ],
                  );
                },
              );
            },
          ),
        ],
      ),
      body: StreamBuilder(
        stream:
databaseRef.child(user.uid).child('profiles').onValue,
        builder: (context, snapshot) {
          if (!snapshot.hasData) {
            return Center(
                child: SpinKitFadingFour(
              color: appcolortheme,
            ));
          }
          if (snapshot.data!.snapshot.value is Map) {
            return ListView.builder(
              itemCount: (snapshot.data!.snapshot.value as
Map).keys.length,
              itemBuilder: (context, index) {
                String key = (snapshot.data!.snapshot.value as Map)
                    .keys
                    .elementAt(index);
                String name =
                    (snapshot.data!.snapshot.value as
```

```
Map)[key]['name'];
                List fingerprints;
                if ((snapshot.data?.snapshot.value as
Map)[key]['fingerprints']
                    is List) {
                  fingerprints = (snapshot.data?.snapshot.value as
Map)[key]
                      ['fingerprints'];
                } else {
                  fingerprints = [];
                }

                return InkWell(
                    onTap: () =>
Navigator.of(context).push(MaterialPageRoute(
                        builder: (BuildContext context) =>
FingerprintPage(
                            profileKey: key, fingerprints:
fingerprints),
                      )),
                    child: Column(
                      children: [
                        Dismissible(
                          key: UniqueKey(),
                          direction: DismissDirection.endToStart,
                          onDismissed: (direction) {
                            // Show a confirmation dialog before
deleting the profile
                            showDialog(
                              context: context,
                              builder: (BuildContext context) {
                                return AlertDialog(
                                  title: Text("Confirm deletion"),
                                  content: Text(
                                      "Are you sure you want to
delete this profile?"),
                                  actions: <Widget>[
                                    TextButton(
                                      child: Text("Cancel"),
                                      onPressed: () {
                                        Navigator.of(context).pop();
                                        setState(() {});
                                      },
                                    ),
                                    TextButton(
                                      child: Text(
                                        "Delete",
                                        style: TextStyle(color:
Colors.red),
                                      ),
                                      onPressed: () {
                                        Navigator.of(context).pop();
                                        _deleteProfile(key);
                                      },
                                    ),
                                  ],
                                );
                              },
                            );
```

```
                        },
                        background: Container(color: Colors.red),
                        secondaryBackground: Container(
                          color: Colors.red,
                          child: Align(
                              alignment: Alignment.centerRight,
                              child: Row(
                                mainAxisAlignment:
MainAxisAlignment.end,
                                children: <Widget>[
                                  Icon(Icons.delete),
                                  Text("Deleting...")
                                ],
                              )),
                        ),
                        child: ListTile(
                          title: Text(name),
                          subtitle: ListView.builder(
                              shrinkWrap: true,
                              itemCount: fingerprints.length,
                              itemBuilder: (context, index) {
                                return Text(fingerprints[index]);
                              }),
                        ),
                      )
                    ],
                  ));
            },
          );
        }

        return Center(
            child: Text(
                "              No profiles \npress '+' to add new
profile"));
      },
    ),
  );
 }
}
```

FingerprintPage.dart:

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_database/firebase_database.dart';
import 'package:flutter/material.dart';
import 'package:flutter_spinkit/flutter_spinkit.dart';
import 'package:lottie/lottie.dart';
import '../widget/AddFingerprintDialog.dart';
import '../widget/constants.dart';
import 'FingerprintStepper.dart';

class FingerprintPage extends StatefulWidget {
  final String profileKey;
  final List fingerprints;

  FingerprintPage({required this.profileKey, required
this.fingerprints});

  @override
  State<FingerprintPage> createState() => _FingerprintPageState();
}

String selectedProfileKey = '';

class _FingerprintPageState extends State<FingerprintPage> {
  GlobalKey<RefreshIndicatorState> _refreshKey =
      GlobalKey<RefreshIndicatorState>();
  int? fingerprint;
  String profileKey = '';

  FirebaseDatabase database = FirebaseDatabase.instance;
  final user = FirebaseAuth.instance.currentUser!;
  DatabaseReference databaseRef = FirebaseDatabase.instance.ref();

  bool _isOptionChanged = false;

  void _changeOption(String profileKey) async {
    DatabaseReference optionRef =
databaseRef.child(user.uid).child('option');
    optionRef.set(1);
    _isOptionChanged = true;
  }

  void _addFingerprint(
      String profileKey, int fingerprint, Function callback) async {
    DatabaseReference profilesRef =
        databaseRef.child(user.uid).child('profiles');
    final DatabaseEvent event = await profilesRef.once();
    final DataSnapshot snapshot = event.snapshot;
    Map<dynamic, dynamic>? profiles = snapshot.value as Map?;
    if (profiles != null) {
      for (var profile in profiles.values) {
        if (profile['fingerprints'] != null) {
          for (var fp in profile['fingerprints'].values) {
            if (fp == fingerprint) {
              //fingerprint already exists
              showDialog(
                context: context,
                builder: (BuildContext context) {
```

```
                  return AlertDialog(
                    title: Text(
                      "Error",
                      style: TextStyle(color: Colors.red),
                    ),
                    content: Text("Fingerprint already exists"),
                    actions: <Widget>[
                      TextButton(
                        child: Text("OK"),
                        onPressed: () {
                          Navigator.of(context).pop();
                        },
                      ),
                    ],
                  );
                },
              );
              return;
            }
          }
        }
      }
    }
    // Generate a unique key for the new fingerprint node
    DatabaseReference fingerprintsRef = databaseRef
        .child(user.uid)
        .child('profiles')
        .child(profileKey)
        .child('fingerprints');
    DatabaseReference fingerprintRef = fingerprintsRef.push();
    // Use the generated key to set the value of the fingerprint id
    fingerprintRef.set(fingerprint);
    databaseRef.child(user.uid).child("id").set(fingerprint);
    var fingerprintKey = fingerprintRef.key;
    callback(fingerprintKey);
  }

  void _deleteFingerprint(String profileKey, String fingerprintKey)
async {
    // Get a reference to the fingerprint node using the profileKey
and fingerprint key
    DatabaseReference fingerprintRef = FirebaseDatabase.instance
        .ref()
        .child(user.uid)
        .child('profiles')
        .child(profileKey)
        .child('fingerprints')
        .child(fingerprintKey);
    // Delete the fingerprint node
    fingerprintRef.remove();
  }

  void _deleteAddedFingerprint(String fingerprintKey) async {
    // Get a reference to the recently added fingerprint node using
the fingerprint key
    DatabaseReference fingerprintRef = FirebaseDatabase.instance
        .ref()
        .child(user.uid)
        .child('profiles')
```

```
        .child(widget.profileKey)
        .child('fingerprints')
        .child(fingerprintKey);
    // Delete the fingerprint node
    fingerprintRef.remove();
  }

  Future<void> _refreshData() async {
    // Code to refresh data from Firebase
    setState(() {});
    //...
    _refreshKey.currentState?.show();
  }

  @override
  void initState() {
    super.initState();
    _refreshData();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Fingerprints'),
        backgroundColor: appcolortheme,
        actions: [
          IconButton(
            icon: Icon(Icons.fingerprint),
            onPressed: () async {
              _changeOption(widget.profileKey);
              // Show the dialog to add the fingerprint ID
              int fingerprint = await showDialog(
                context: context,
                builder: (context) => AddFingerprintDialog(),
              );

              // If the user entered a fingerprint ID
              _addFingerprint(widget.profileKey, fingerprint,
(fingerprintKey) {
                Navigator.push(
                  context,
                  MaterialPageRoute(
                      builder: (context) => FingerprintStepper(
                          widget.profileKey,
                          _deleteAddedFingerprint,
                          fingerprintKey)),
                );
              });
            },
          ),
        ],
      ),
      body: RefreshIndicator(
        key: _refreshKey,
        onRefresh: _refreshData,
        child: StreamBuilder(
          stream: databaseRef
              .child(user.uid)
```

```
              .child('profiles')
              .child(widget.profileKey)
              .child('fingerprints')
              .onValue,
          builder: (context, snapshot) {
            if (snapshot.hasData) {
              var value = snapshot.data?.snapshot.value;
              if (value is Map) {
                Map fingerprints = value;
                if (fingerprints.keys.length > 0) {
                  return ListView.builder(
                    itemCount: fingerprints.length,
                    itemBuilder: (BuildContext context, int index) {
                      return Dismissible(
                        key: UniqueKey(),
                        direction: DismissDirection.endToStart,
                        onDismissed: (direction) {
                          showDialog(
                            context: context,
                            builder: (BuildContext context) {
                              return AlertDialog(
                                title: Text("Confirm deletion"),
                                content: Text(
                                    "Are you sure you want to delete
this fingerprint?"),
                                actions: <Widget>[
                                  TextButton(
                                    child: Text("Cancel"),
                                    onPressed: () {
                                      Navigator.of(context).pop();
                                      setState(() {});
                                    },
                                  ),
                                  TextButton(
                                    child: Text("Delete",
                                        style: TextStyle(color:
Colors.red)),
                                    onPressed: () {
                                      Navigator.of(context).pop();

_deleteFingerprint(widget.profileKey,

fingerprints.keys.elementAt(index));
                                    },
                                  ),
                                ],
                              );
                            },
                          );
                        },
                        background: Container(color: Colors.red),
                        secondaryBackground: Container(
                          color: Colors.red,
                          child: Align(
                              alignment: Alignment.centerRight,
                              child: Row(
                                mainAxisAlignment:
MainAxisAlignment.end,
                                children: <Widget>[
```

```
                                    Icon(Icons.delete),
                                    Text("Deleting...")
                                  ],
                                )),
                            ),
                            child: ListTile(
                              title: Text(

fingerprints.values.elementAt(index).toString()),
                            ),
                          );
                        },
                      );
                    } else {
                      return Center(
                          child: SpinKitPouringHourGlassRefined(
                        color: appcolortheme,
                      ));
                    }
                  } else {
                    return Center(
                      child: SingleChildScrollView(
                        child: Column(
                          mainAxisAlignment: MainAxisAlignment.start,
                          crossAxisAlignment: CrossAxisAlignment.center,
                          children: <Widget>[
                            Lottie.asset('assets/nofingerprints.json'),
                            SizedBox(
                              height: 30,
                            ),
                            Text(
                              "No Fingerprints Saved",
                              style: TextStyle(fontSize: 25),
                            ),
                          ],
                        ),
                      ),
                    );
                  }
                } else {
                  return Center(
                      child: SpinKitFadingFour(
                    color: appcolortheme,
                  ));
                }
              },
            ),
          ),
        );
    }
}
```

AddFingerprintDialog.dart:

```
import 'package:cypherflutter/navpages/profile_page.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_database/firebase_database.dart';
import 'package:flutter/material.dart';

class AddFingerprintDialog extends StatefulWidget {
  @override
  _AddFingerprintDialogState createState() =>
_AddFingerprintDialogState();
}

class _AddFingerprintDialogState extends State<AddFingerprintDialog>
{
  FirebaseDatabase database = FirebaseDatabase.instance;
  final user = FirebaseAuth.instance.currentUser!;
  DatabaseReference databaseRef = FirebaseDatabase.instance.ref();
  int? _fingerprintId;
  String _errorText = "";

  void _changeOptionBack(String profileKey) async {
    DatabaseReference optionRef =
databaseRef.child(user.uid).child('option');
    optionRef.set(2);
  }

  @override
  Widget build(BuildContext context) {
    return WillPopScope(
      onWillPop: () async {
        _changeOptionBack(selectedProfileKey);
        return true;
      },
      child: AlertDialog(
        title: Text('Add Fingerprint'),
        content: Column(
          mainAxisSize: MainAxisSize.min,
          children: <Widget>[
            TextField(
              keyboardType: TextInputType.number,
              onChanged: (value) {
                if (int.tryParse(value) != null &&
                    int.parse(value) >= 1 &&
                    int.parse(value) <= 127) {
                  _fingerprintId = int.parse(value);
                  _errorText = "";
                } else {
                  _fingerprintId = null;
                  _errorText = " Number between 1 & 127";
                }
                setState(() {});
              },
              decoration: InputDecoration(
                focusedBorder: OutlineInputBorder(
                  borderSide: BorderSide(color: Colors.lightGreen),
                ),
                border: OutlineInputBorder(),
```

```
                labelText: 'Enter Fingerprint ID',
                labelStyle: TextStyle(color: Colors.black),
                errorText: _errorText,
              ),
            ),
          ],
        ),
        actions: <Widget>[
          TextButton(
            child: Text('Cancel'),
            onPressed: () {
              _changeOptionBack(selectedProfileKey);
              Navigator.of(context).pop();
            },
          ),
          TextButton(
            child: Text(
              'Add',
              style: TextStyle(color: Colors.green),
            ),
            onPressed: () {
              if (_fingerprintId != null) {
                Navigator.of(context).pop(_fingerprintId);
              }
            },
          ),
        ],
      ),
    );
  }
}
```

FingerprintStepper.dart:

```
import 'dart:math';
import 'package:cypherflutter/navpages/profile_page.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_database/firebase_database.dart';
import 'package:flutter/material.dart';
import 'package:flutter_spinkit/flutter_spinkit.dart';
import 'package:lottie/lottie.dart';

import '../widget/constants.dart';

class FingerprintStepper extends StatefulWidget {
  final String profileKey;
  final Function deleteAddedFingerprint;
  final String fingerprintKey;

  FingerprintStepper(
      this.profileKey, this.deleteAddedFingerprint,
this.fingerprintKey);

  @override
  _FingerprintStepperState createState() =>
_FingerprintStepperState();
}

class _FingerprintStepperState extends State<FingerprintStepper> {
  void _deleteAddedFingerprint() {
    widget.deleteAddedFingerprint(widget.fingerprintKey);
  }

  int currentStep = 0;
  bool _isOptionChanged = true;

  FirebaseDatabase database = FirebaseDatabase.instance;
  final user = FirebaseAuth.instance.currentUser!;
  DatabaseReference databaseRef = FirebaseDatabase.instance.ref();

  void _changeOptionBack(String profileKey) async {
    DatabaseReference optionRef =
databaseRef.child(user.uid).child('option');
    optionRef.set(2);
  }

  @override
  void initState() {
    super.initState();
    // Listen for changes to the "steps" node under the user's uid
    databaseRef.child(user.uid).child("steps").set(0);

databaseRef.child(user.uid).child("steps").onValue.listen((event) {
      var step = int.parse(event.snapshot.value.toString());
      setState(() {
        currentStep = min(max(step - 1, 0), getSteps().length - 1);
      });
    });
  }

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

```
    return WillPopScope(
      onWillPop: () async {
        _deleteAddedFingerprint();
        if (_isOptionChanged) {
          _changeOptionBack(selectedProfileKey);
        }
        return true;
      },
      child: Scaffold(
        appBar: AppBar(
          title: Text('Setup fingerprint'),
          backgroundColor: appcolortheme,
        ),
        body: StreamBuilder(
            stream:
databaseRef.child(user.uid).child("steps").onValue,
            builder: (BuildContext context, AsyncSnapshot snapshot)
{
              if (snapshot.hasData) {
                var value = snapshot.data.snapshot.value;
                if (value == null) {
                  print("value is null");
                  return Center(
                      child: SpinKitFadingCircle(
                    color: Colors.red,
                  ));
                }
                try {
                  var step = int.parse(value.toString());
                  if (step > 0) {
                    return Theme(
                      data: Theme.of(context).copyWith(
                        colorScheme: ColorScheme.light(primary:
Colors.green),
                      ),
                      child: Stepper(
                        type: StepperType.vertical,
                        steps: getSteps(),
                        currentStep: currentStep,
                        controlsBuilder: (context, controller) {
                          return const SizedBox.shrink();
                        },
                      ),
                    );
                  } else {
                    return Center(
                      child: Column(
                        mainAxisAlignment: MainAxisAlignment.start,
                        children: <Widget>[
                          SizedBox(
                            height: 60,
                          ),
                          Lottie.asset('assets/stepper.json'),
                          Text(
                            "Please Wait...",
                            style: TextStyle(fontSize: 30),
                          ),
                        ],
                      ),
```

```
                    );
                  }
                } catch (e) {
                  print("Not able to parse value: $value to int");
                  return Center(
                      child: SpinKitFadingCircle(
                    color: appcolortheme,
                  ));
                }
              } else {
                return Center(
                    child: SpinKitFadingFour(
                  color: appcolortheme,
                ));
              }
            }),
      ),
    );
  }

  List<Step> getSteps() => [
        Step(
          state: currentStep > 0 ? StepState.complete :
StepState.indexed,
          isActive: currentStep >= 0,
          title: Text('Scan Finger'),
          content: Container(),
        ),
        Step(
          state: currentStep > 1 ? StepState.complete :
StepState.indexed,
          isActive: currentStep >= 1,
          title: Text('Remove finger'),
          content: Container(),
        ),
        Step(
          state: currentStep > 2 ? StepState.complete :
StepState.indexed,
          isActive: currentStep >= 2,
          title: Text('Scan Finger Again'),
          content: Container(),
        ),
        Step(
          state: currentStep > 3 ? StepState.complete :
StepState.indexed,
          isActive: currentStep >= 3,
          title: Text('Success / Failure'),
          content: StreamBuilder(
              stream:
databaseRef.child(user.uid).child("steps").onValue,
              builder: (BuildContext context, AsyncSnapshot
snapshot) {
                if (snapshot.hasData) {
                  var value = snapshot.data.snapshot.value;
                  if (value == null) {
                    return Container();
                  }
                  try {
                    var step = int.parse(value.toString());
```

```
                    return Container(
                      child: Column(
                        children: [
                          Visibility(
                            visible: step == 4,
                            child: AlertDialog(
                              title: Text("Success"),
                              content: Text("Fingerprint Saved
Successfully!"),
                              actions: [
                                TextButton(
                                    onPressed: () {
                                      Navigator.pop(context);
                                      if (_isOptionChanged) {

_changeOptionBack(selectedProfileKey);
                                      }
                                      ;
                                    },
                                    child: Text("OK"))
                              ],
                            ),
                          ),
                          Visibility(
                            visible: step != 4,
                            child: SafeArea(
                              child: AlertDialog(
                                title: Text(
                                  "Oops... Failed",
                                  style: TextStyle(color:
Colors.red),
                                ),
                                content: Text("   Please try
again!"),
                                actions: [
                                  TextButton(
                                    child: Text("OK"),
                                    onPressed: () {
                                      _deleteAddedFingerprint();
                                      Navigator.pop(context);
                                      if (_isOptionChanged) {

_changeOptionBack(selectedProfileKey);
                                      }
                                    },
                                  ),
                                ],
                                elevation: 24,
                                icon: Icon(Icons.cancel),
                              ),
                            ),
                          ),
                        ],
                      ),
                    );
                  } catch (e) {
                    return Container();
                  }
                } else {
```

```
                    return Container();
                  }
                ),
            ),
        ];
}
```

history_page.dart:

```
import 'package:cypherflutter/widget/constants.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_database/firebase_database.dart';
import 'package:flutter/material.dart';
import 'package:flutter_spinkit/flutter_spinkit.dart';
import 'package:intl/intl.dart';
import 'package:lottie/lottie.dart';

class HistoryPage extends StatefulWidget {
  @override
  _HistoryPageState createState() => _HistoryPageState();
}

class _HistoryPageState extends State<HistoryPage> {
  FirebaseDatabase database = FirebaseDatabase.instance;
  final user = FirebaseAuth.instance.currentUser!;
  DatabaseReference databaseRef = FirebaseDatabase.instance.ref();
  late String _profileName;
  late String _dateTime;
  int? _savedID;

  void _activateListeners() {
    // Listen for changes to the savedID value

databaseRef.child(user.uid).child("savedID").onValue.listen((event)
{
      final DataSnapshot snapshot = event.snapshot;
      _savedID = snapshot.value as int;
      _updateHistory();
    });
  }

  Future<void> _updateHistory() async {
    // Check if the history node exists
    final DataSnapshot historySnapshot =
        await databaseRef.child(user.uid).child("history").get();
    if (!historySnapshot.exists) {
      // If the history node does not exist, create it
      databaseRef.child(user.uid).child("history").set({});
    }
    // Check all fingerprints under all profiles until a match is
found
    final DataSnapshot snapshot =
        await databaseRef.child(user.uid).child("profiles").get();
    final Map<dynamic, dynamic> profiles =
        (snapshot.value as Map<dynamic, dynamic>);
    for (final dynamic profile in profiles.values) {
      if (profile['fingerprints'] != null) {
        for (final dynamic fp in profile['fingerprints'].values) {
          if (fp == _savedID) {
            _profileName = profile['name'];
            final DateTime currentTime = DateTime.now();
            // format the date and time
            _dateTime = DateFormat('dd-MM-yyyy
HH:mm').format(currentTime);
            // add the profile name, date and time to the Firebase
Realtime Database history collection
```

```
databaseRef.child(user.uid).child("history").push().set({
              'profileName': _profileName,
              'dateTime': _dateTime,
            });
            return;
          }
        }
      }
    }
  }

  @override
  void initState() {
    super.initState();
    _activateListeners();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('History'),
        backgroundColor: appcolortheme,
      ),
      body: StreamBuilder<DatabaseEvent>(
        stream:
databaseRef.child(user.uid).child("history").onValue,
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return Center(
                child: SpinKitFadingFour(
              color: appcolortheme,
            ));
          }
          if (snapshot.hasData && snapshot.data?.snapshot.value !=
null) {
            final List history =
                (snapshot.data?.snapshot.value as Map<dynamic,
dynamic>)
                    .values
                    .toList();
            if (history.length > 0) {
              history.sort((a, b) =>
b['dateTime'].compareTo(a['dateTime']));
              return ListView.builder(
                itemCount: history.length,
                itemBuilder: (context, index) {
                  final dynamic item = history[index];
                  return ListTile(
                    title: Text(item['profileName'] + ' came home'),
                    subtitle: Text(item['dateTime']),
                  );
                },
              );
            } else {
              return Center(
                  child: SpinKitPouringHourGlassRefined(
                color: appcolortheme,
```

```
              ));
            }
          } else {
            return Center(
              child: Column(
                mainAxisAlignment: MainAxisAlignment.start,
                crossAxisAlignment: CrossAxisAlignment.center,
                children: <Widget>[
                  SizedBox(
                    height: 190,
                  ),
                  Lottie.asset('assets/nohistory.json'),
                  SizedBox(
                    height: 20,
                  ),
                  Text(
                    "No history yet",
                    style: TextStyle(fontSize: 25),
                  ),
                ],
              ),
            );
          }
        },
      ),
    );
  }
}
```

settings_page.dart:

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

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

import '../widget/WiFiSettingsPage.dart';
import '../widget/constants.dart';

class settingspage extends StatefulWidget {
  const settingspage({Key? key}) : super(key: key);

  @override
  State<settingspage> createState() => _settingspageState();
}

class _settingspageState extends State<settingspage> {
  final user = FirebaseAuth.instance.currentUser!;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Settings'),
        backgroundColor: appcolortheme,
      ),
      body: Padding(
          padding: EdgeInsets.all(27),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Container(
                padding: EdgeInsets.all(20),
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Text(
                      'Account:',
                      style: TextStyle(
                        fontSize: 20,
                        color: Colors.black,
                        fontWeight: FontWeight.bold,
                      ),
                    ),
                    SizedBox(height: 10),
                    Text(
                      user.email.toString(),
                      style: TextStyle(
                        fontSize: 18,
                        color: Colors.grey[600],
                      ),
                    )
                  ],
                ),
              ),
              Card(
                child: ListTile(
                  leading: Icon(Icons.wifi_protected_setup_rounded),
```

```
                  title: Text("Wi-Fi Connection"),
                  onTap: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                          builder: (context) => WiFiSettingsPage()),
                    );
                  },
                ),
              ),
              SizedBox(height: 30),
              Card(
                child: ListTile(
                  leading: Icon(Icons.phone, color: Colors.red),
                  title: Text('Emergency Call',
                      style: TextStyle(color: Colors.red)),
                  onTap: () async {
                    var url = Uri.parse("tel:999");
                    if (await canLaunchUrl(url)) {
                      await launchUrl(url);
                    } else {
                      throw 'Could not launch $url';
                    }
                  },
                ),
              ),
              SizedBox(height: 30),
              Card(
                child: ListTile(
                  leading: Icon(Icons.logout, color: Colors.green,
size: 32),
                  title: Text(
                    'Sign Out',
                    style: TextStyle(color: Colors.green, fontSize:
24),
                  ),
                  onTap: () => FirebaseAuth.instance.signOut(),
                ),
              ),
            ],
          )),
    );
  }
}
```

WifiSettingsPage.dart:

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

import 'constants.dart';

class WiFiSettingsPage extends StatefulWidget {
  @override
  _WiFiSettingsPageState createState() => _WiFiSettingsPageState();
}

class _WiFiSettingsPageState extends State<WiFiSettingsPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Wi-Fi Settings"),
        backgroundColor: appcolortheme,
      ),
      body: Center(
        child: Card(
          child: Padding(
            padding: EdgeInsets.all(16.0),
            child: Column(
              children: <Widget>[
                Text(
                  "Connecting Your System to the Internet",
                  style: TextStyle(fontSize: 22),
                ),
                SizedBox(height: 20),
                Text(
                    "1. Turn on your system. The server that
connects to the internet will automatically start working."),
                SizedBox(height: 10),
                Align(
                  alignment: Alignment.centerLeft,
                  child: Text("2. Go to your device's WiFi
settings."),
                ),
                SizedBox(height: 10),
                Card(
                  color: appcolortheme,
                  child: ListTile(
                    leading: Icon(
                      Icons.wifi,
                      color: Colors.white,
                    ),
                    title: Text(
                      'Wi-fi Settings',
                      style: TextStyle(color: Colors.white),
                    ),
                    onTap: () {
                      OpenSettings.openWIFISetting();
                    },
                  ),
                ),
                SizedBox(height: 20),
                Text(
```

```
                    "3. Look for a WiFi network named 'Cypher' and
enter the password 'password' or scan the QR code on the back of the
box using your phone's QR scanner."),
                SizedBox(height: 10),
                Text(
                    "4. Press 'Connect.' A server window will open
on your phone."),
                SizedBox(height: 10),
                Text(
                    "5. You will see a list of available networks in
the area. Select the network you want your system to connect to by
clicking on it."),
                SizedBox(height: 10),
                Text(
                    "6. Enter the password for the selected network
and press 'Save'."),
                SizedBox(height: 10),
                Align(
                  alignment: Alignment.centerLeft,
                  child: Text(
                      "7. Your system will now be connected to the
internet."),
                ),
                SizedBox(height: 20),
                Align(
                    alignment: Alignment.centerLeft,
                    child: Text(
                      "Note:",
                      style: TextStyle(fontWeight: FontWeight.bold),
                    )),
                Text(
                    "If you want to change the network your system
is connected to, turn off the previously saved network. This will
disconnect the system from the network and allow you to connect to a
different one by following the steps above."),
                SizedBox(height: 16.0),
              ],
            ),
          ),
        ),
      ),
    );
  }
}
```

pubspec.yaml:

```yaml
name: cypherflutter
description: A new Flutter project.

publish_to:


version: 1.0.0+1

environment:
  sdk: '>=2.18.5 <3.0.0'


dependencies:
  flutter:
    sdk: flutter


  cupertino_icons: ^1.0.2
  google_nav_bar: ^5.0.6
  line_icons: ^2.0.1
  firebase_auth: ^4.2.1
  flutter_login: ^4.1.1
  quickalert: ^1.0.1
  firebase_core: ^2.4.0
  firebase_database: ^10.0.7
  font_awesome_flutter: ^10.3.0
  local_auth: ^2.1.3
  flutter_secure_storage: ^7.0.1
  open_settings: ^2.0.2
  awesome_notifications: ^0.7.4+1
  intl: ^0.17.0
  url_launcher: ^6.1.8
  flutter_spinkit: ^5.1.0
  lottie: ^2.2.0


dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_launcher_icons: ^0.11.0

flutter_icons:
  android: true
  ios: true
  image_path: "assets/logo.png"
  adaptive_icon_background: "#12CDEC"
  adaptive_icon_foreground: "assets/logo_foreground.png"
  flutter_lints: ^2.0.0

flutter:


  uses-material-design: true

  assets:
    - assets/
```

firebase_options.dart:

```
// File generated by FlutterFire CLI.
// ignore_for_file: lines_longer_than_80_chars,
avoid_classes_with_only_static_members
import 'package:firebase_core/firebase_core.dart' show
FirebaseOptions;
import 'package:flutter/foundation.dart'
    show defaultTargetPlatform, kIsWeb, TargetPlatform;

/// Default [FirebaseOptions] for use with your Firebase apps.
///
/// Example:
/// ```dart
/// import 'firebase_options.dart';
/// // ...
/// await Firebase.initializeApp(
///   options: DefaultFirebaseOptions.currentPlatform,
/// );
/// ```
class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    if (kIsWeb) {
      throw UnsupportedError(
        'DefaultFirebaseOptions have not been configured for web - '
        'you can reconfigure this by running the FlutterFire CLI
again.',
      );
    }
    switch (defaultTargetPlatform) {
      case TargetPlatform.android:
        return android;
      case TargetPlatform.iOS:
        return ios;
      case TargetPlatform.macOS:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for macos
- '
          'you can reconfigure this by running the FlutterFire CLI
again.',
        );
      case TargetPlatform.windows:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for
windows - '
          'you can reconfigure this by running the FlutterFire CLI
again.',
        );
      case TargetPlatform.linux:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for linux
- '
          'you can reconfigure this by running the FlutterFire CLI
again.',
        );
      default:
        throw UnsupportedError(
          'DefaultFirebaseOptions are not supported for this
platform.',
```

```
        );
    }
  }

  static const FirebaseOptions android = FirebaseOptions(
    apiKey: 'AIzaSyBj9Oq99OajhNrfi00W8Vbrv05p63WUWDY',
    appId: '1:777164042128:android:abbc4713868fb549c14d15',
    messagingSenderId: '777164042128',
    projectId: 'cypherflutter-4e3d3',
    databaseURL: 'https://cypherflutter-4e3d3-default-
rtdb.firebaseio.com',
    storageBucket: 'cypherflutter-4e3d3.appspot.com',
  );

  static const FirebaseOptions ios = FirebaseOptions(
    apiKey: 'AIzaSyD-Y7CAB-ZvEUHIyvZmg6fVbj9S3Wmmjvo',
    appId: '1:777164042128:ios:57adc71cfbda3467c14d15',
    messagingSenderId: '777164042128',
    projectId: 'cypherflutter-4e3d3',
    databaseURL: 'https://cypherflutter-4e3d3-default-
rtdb.firebaseio.com',
    storageBucket: 'cypherflutter-4e3d3.appspot.com',
    iosClientId: '777164042128-
9tajqn0ciamkeiheattoipr39ul8tg7r.apps.googleusercontent.com',
    iosBundleId: 'com.example.cypherflutter',
  );
}
```

Arduino code:

```
#include <Arduino.h>

#include <ESP8266WiFi.h>

#include <Adafruit_Fingerprint.h>

#include <SoftwareSerial.h>

#include <Firebase_ESP_Client.h>

#include <Wire.h>

#include <WiFiManager.h>

#include <ctime>

// Provide the token generation process info.

#include "addons/TokenHelper.h"
```

33 | P a g e

```
// Provide the RTDB payload printing info and other helper functions.

#include "addons/RTDBHelper.h"

#define FIREBASE_HOST "https://cypherflutter-4e3d3-default    rtdb.firebaseio.com" //URL
ADDRESS OF OUR FIREBASE.

https://cypherflutter-4e3d3-default-rtdb.firebaseio.com/

#define FIREBASE_AUTH "lCBkT6pZ3JAndHvQy3KUg0w3zrV9XduEPSjGHxo5" // THE

SECRET CODE OF OUR DATABASE

// #define WIFI_SSID "ahmed"

// #define WIFI_PASSWORD "Ahmed1937"

#define API_KEY "AIzaSyBj9Oq99OajhNrfi00W8Vbrv05p63WUWDY"

#define USER_EMAIL "1181102334@student.mmu.edu.my"

#define USER_PASSWORD "password"

#define trigger_pin 2

#define Echo_pin 14

int buzzerPin = D6;
```

```
int motion = 4; // Digital pin D7

// int Status = 13; // Digital pin D6

const int SLEEP_TIME = 1;

// defines variables

long duration;

int distance;

// firebase variables

FirebaseData fbdo;

FirebaseAuth auth;

FirebaseConfig config;

volatile int finger_status = -1;

SoftwareSerial mySerial(13, 15, false);

Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

uint8_t id;

uint8_t readnumber(void) {

uint8_t num = 0;

while (num == 0) {

while (! Serial.available());

num = Serial.parseInt();

}

return num;

}
```



```
String uid;

void scan();

void check_in();
```

```
int test();
int getFingerprintIDez();
void initWiFi() {
WiFiManager wm;
//
// // Automatically connect using saved credentials,
// // if connection fails, it starts an access point with the
specified name ( "AutoConnectAP"),
// // if empty will auto generate SSID, if password is blank
it will be anonymous AP (wm.autoConnect())
// // then goes into a blocking loop awaiting configuration
and will return success result
bool res = wm.autoConnect("Cypher","password"); // password
protected ap
if(!res) {
Serial.println("Failed to connect");
}
else {
//if you get here you have connected to the WiFi
Serial.println("connected...yeey :)");
}
Serial.println(wm.getWiFiSSID());
Serial.println(wm.getWiFiPass());
}
void Firebaseconnection(){
config.api_key = API_KEY;
```

```
config.database_url = FIREBASE_HOST;
// Assign the user sign in credentials
auth.user.email = USER_EMAIL;
auth.user.password = USER_PASSWORD;
Firebase.reconnectWiFi(true);
fbdo.setResponseSize(4096);
// Assign the callback function for the long running token generation
task
config.token_status_callback = tokenStatusCallback; //see
addons/TokenHelper.h
// Assign the maximum retry of token generation
config.max_token_generation_retry = 5;
// Initialize the library with the Firebase authen and config
Firebase.begin(&config, &auth);
// Getting the user UID might take a few seconds
35 | P a g e
Serial.println("Getting User UID");
while ((auth.token.uid) == "") {
Serial.print('.');
delay(500);
}
// Print user UID
uid = auth.token.uid.c_str();
Serial.println("User UID: ");
Serial.print(uid);
}
```

```
void setup() {

Serial.begin(9600);

pinMode(trigger_pin, OUTPUT); // configure the trigger_pin(D9) as an

Output

pinMode(motion, INPUT); // declare sensor as input

// pinMode(Status, OUTPUT); // declare LED as

pinMode(Echo_pin, INPUT); // configure the Echo_pin(D11) as an Input

pinMode(D1, OUTPUT);

pinMode(buzzerPin, OUTPUT);

Serial.begin(9600);

// put your setup code here, to run once:

while (!Serial);

delay(100);

Serial.println("\n\nAdafruit Fingerprint sensor enrollment");

finger.begin(57600);

// finger.verifyPassword();

if (finger.verifyPassword()) {

Serial.println("Found fingerprint sensor!");

} else {

Serial.println("Did not find fingerprint sensor :(");

while (1) { delay(1); }

}

finger.getTemplateCount();

Serial.print("Sensor contains "); Serial.print(finger.templateCount);

Serial.println(" templates");

}
```

```
void loop() {
```



```
digitalWrite(trigger_pin, LOW); //set trigger signal low for 2us
delayMicroseconds(2);
// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigger_pin, HIGH); // make trigger pin active high
delayMicroseconds(10); // wait for 10 microseconds
digitalWrite(trigger_pin, LOW); // make trigger pin active low
duration = pulseIn(Echo_pin, HIGH); // save time duration value in
"duration variable
distance= duration*0.034/2; //Convert pulse duration into distance
long state = digitalRead(motion);
Serial.println(motion);
if ( distance < 30 && distance > 2 && state == HIGH ){
Serial.println("Waiting for valid finger");
check_in();
initWiFi();
Firebaseconnection();
Serial.println("Motion detected!");
Serial.print("Distance: ");
Serial.print(distance);
Serial.println(" cm");
Firebase.RTDB.set(&fbdo, uid + "/distance", distance);
Firebase.RTDB.set(&fbdo, uid + "/motion", true);
state = fbdo.to<bool>();
delay(5000);
```

```
Serial.println("Choose 1 for scan, 2 for check-in.");

delay(1000);

Firebase.RTDB.get(&fbdo, uid + "/option");

int option = fbdo.to<int>();

// Firebase.RTDB.get(&fbdo, uid + "/random");

Firebase.RTDB.get(&fbdo, uid + "/doorLocked" );

bool doorLocked = fbdo.to<bool>();

Serial.println(option);
```



```
if ( option == 1 ){

switch(option)

{

case 1:

scan();

test();

break;

// case 2:

}

}

else if ( option == 2 && doorLocked == false ){

Serial.println("opening door");

digitalWrite(D1, HIGH);

delay(5000);

digitalWrite(D1, LOW);

delay(500);

Firebase.RTDB.set(&fbdo, uid + "/doorLocked", true );
```

```
}
Firebase.RTDB.set(&fbdo, uid + "/motion", false);
state = fbdo.to<bool>();
}
else {
Serial.println("Sleep");
ESP.deepSleep( 1e6);
}
}
void scan()
{
Serial.println("Ready to enroll a fingerprint!");
Firebase.RTDB.get(&fbdo, uid + "/id");
id = fbdo.to<int>();
Serial.println(id);
if (id == 0) {
return;
}
Serial.print("Enrolling ID #");
Serial.println(id);
38 | P a g e
uint8_t getFingerprintEnroll();
}
void check_in()
{
finger_status = getFingerprintIDez();
```

```
if (finger_status!=-1 and finger_status!=-2){
Serial.print("Match");
}
else{
if (finger_status==-2){
Serial.print("Not Match");
initWiFi();
Firebaseconnection();
Firebase.RTDB.setString(&fbdo, uid + "/finger_status/", "Not Match"
);
delay(5000);
}
}
delay(500); //don't ned to run this at full speed.
}
int test(){
int p = -1;
Serial.print("Waiting for valid finger to enroll as #");
Serial.println(id);
Firebase.RTDB.set(&fbdo, uid + "/steps", 1);
delay(5000);
while (p != FINGERPRINT_OK) {
p = finger.getImage();
if (FINGERPRINT_NOFINGER){
break;
```

```
}
switch (p) {
case FINGERPRINT_OK:
Serial.println("Image taken");
break;
case FINGERPRINT_PACKETRECIEVEERR:
Serial.println("Communication error");
Firebase.RTDB.set(&fbdo, uid + "/steps", 5);
break;
case FINGERPRINT_IMAGEFAIL:
Serial.println("Imaging error");
Firebase.RTDB.set(&fbdo, uid + "/steps", 5);
break;
default:
Serial.println("Unknown error");
Firebase.RTDB.set(&fbdo, uid + "/steps", 5);
break;
}
}
// OK success!
p = finger.image2Tz(1);
switch (p) {
case FINGERPRINT_OK:
Serial.println("Image converted");
break;
case FINGERPRINT_IMAGEMESS:
```

```
Serial.println("Image too messy");
Firebase.RTDB.set(&fbdo, uid + "/steps", 5);
return p;
case FINGERPRINT_PACKETRECIEVEERR:
Serial.println("Communication error");
Firebase.RTDB.set(&fbdo, uid + "/steps", 5);
return p;
case FINGERPRINT_FEATUREFAIL:
Serial.println("Could not find fingerprint features");
Firebase.RTDB.set(&fbdo, uid + "/steps", 5);
return p;
case FINGERPRINT_INVALIDIMAGE:
Serial.println("Could not find fingerprint features");
Firebase.RTDB.set(&fbdo, uid + "/steps", 5);
return p;
default:
Serial.println("Unknown error");
Firebase.RTDB.set(&fbdo, uid + "/steps", 5);
return p;
}
Serial.println("Remove finger");
Firebase.RTDB.set(&fbdo, uid + "/steps", 2);
40 | P a g e
delay(10000);
p = 0;
while (p != FINGERPRINT_NOFINGER) {
```

```
p = finger.getImage();
}
Serial.print("ID "); Serial.println(id);
p = -1;
Serial.println("Place same finger again");
Firebase.RTDB.set(&fbdo, uid + "/steps", 3);
delay(10000);
while (p != FINGERPRINT_OK) {
p = finger.getImage();
if (FINGERPRINT_NOFINGER){
break;
}
switch (p) {
case FINGERPRINT_OK:
Serial.println("Image taken");
break;
case FINGERPRINT_PACKETRECIEVEERR:
Serial.println("Communication error");
Firebase.RTDB.set(&fbdo, uid + "/steps", 5);
break;
case FINGERPRINT_IMAGEFAIL:
Serial.println("Imaging error");
Firebase.RTDB.set(&fbdo, uid + "/steps", 5);
break;
default:
Serial.println("Unknown error");
```

```
Firebase.RTDB.set(&fbdo, uid + "/steps", 5);
break;
}
}
// OK success!
p = finger.image2Tz(2);
switch (p) {
case FINGERPRINT_OK:
Serial.println("Image converted");
break;
case FINGERPRINT_IMAGEMESS:
Serial.println("Image too messy");
Firebase.RTDB.set(&fbdo, uid + "/steps", 5);
return p;
case FINGERPRINT_PACKETRECIEVEERR:
Serial.println("Communication error");
Firebase.RTDB.set(&fbdo, uid + "/steps", 5);
return p;
```



```
case FINGERPRINT_FEATUREFAIL:
Serial.println("Could not find fingerprint features");
Firebase.RTDB.set(&fbdo, uid + "/steps", 5);
return p;
case FINGERPRINT_INVALIDIMAGE:
Serial.println("Could not find fingerprint features");
Firebase.RTDB.set(&fbdo, uid + "/steps", 5);
```

```
return p;
default:
Serial.println("Unknown error");
Firebase.RTDB.set(&fbdo, uid + "/steps", 5);
return p;
}
// OK converted!
Serial.print("Creating model for #"); Serial.println(id);
p = finger.createModel();
if (p == FINGERPRINT_OK) {
Serial.println("Prints matched!");
} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
Serial.println("Communication error");
Firebase.RTDB.set(&fbdo, uid + "/steps", 5);
return p;
} else if (p == FINGERPRINT_ENROLLMISMATCH) {
Serial.println("Fingerprints did not match");
Firebase.RTDB.set(&fbdo, uid + "/steps", 5);
return p;
} else {
Serial.println("Unknown error");
Firebase.RTDB.set(&fbdo, uid + "/steps", 5);
return p;
}
Serial.print("ID "); Serial.println(id);
p = finger.storeModel(id);
```

```
if (p == FINGERPRINT_OK) {

Serial.println("Stored!");

Firebase.RTDB.set(&fbdo, uid + "/steps", 4);

return p;

} else if (p == FINGERPRINT_PACKETRECIEVEERR) {

Serial.println("Communication error");

Firebase.RTDB.set(&fbdo, uid + "/steps", 5);

return p;

} else if (p == FINGERPRINT_BADLOCATION) {

Serial.println("Could not store in that location");

Firebase.RTDB.set(&fbdo, uid + "/steps", 5);

return p;

} else if (p == FINGERPRINT_FLASHERR) {

Serial.println("Error writing to flash");

Firebase.RTDB.set(&fbdo, uid + "/steps", 5);
```



```
return p;

} else {

Serial.println("Unknown error");

Firebase.RTDB.set(&fbdo, uid + "/steps", 5);

return p;

}

}

//check_in

// returns -1 if failed, otherwise returns ID #

int getFingerprintIDez() {
```

```
unsigned long time;

time = millis();

Serial.println(time);

uint8_t p = finger.getImage();

Serial.println("3 seconds to use finger print");

while (p != FINGERPRINT_OK && (millis() - time < 10000) ) {

p = finger.getImage();

switch (p) {

case FINGERPRINT_OK:

Serial.println("Image taken");

break;

case FINGERPRINT_NOFINGER:

break;

case FINGERPRINT_PACKETRECIEVEERR:

Serial.println("Communication error");

break;

case FINGERPRINT_IMAGEFAIL:

Serial.println("Imaging error");

break;

default:

Serial.println("Unknown error");

break;

}

}

if(p == FINGERPRINT_NOFINGER){

digitalWrite(buzzerPin, HIGH);
```

```
delay(1000);

digitalWrite(buzzerPin, LOW);

43 | P a g e

}

if (p!=2){

Serial.println(p);

}

if (p != FINGERPRINT_OK) return -1;

p = finger.image2Tz();

if (p!=2){

Serial.println(p);

}

if (p != FINGERPRINT_OK) return -1;

p = finger.fingerFastSearch();

if (p != FINGERPRINT_OK) return -2;

// found a match!

Serial.print("Found ID #"); Serial.print(finger.fingerID);

digitalWrite(D1, HIGH);

delay(5000);

digitalWrite(D1, LOW);

initWiFi();

Firebaseconnection();

Firebase.RTDB.set(&fbdo, uid + "/savedID", finger.fingerID);

finger.fingerID = fbdo.to<int>();

Firebase.RTDB.setString(&fbdo, uid + "/finger_status/", "Match");

Firebase.RTDB.set(&fbdo, uid + "/motion", true);
```

```
delay(10000);

Firebase.RTDB.set(&fbdo, uid + "/savedID", 0);

Serial.print(distance);

Firebase.RTDB.set(&fbdo, uid + "/distance", distance);

Firebase.RTDB.set(&fbdo, uid + "/motion", false);

return finger.fingerID;

}
```