

## **ABSTRACT**

Internet access is one of the most crucial things nowadays. However, plenty of people are having excess cellular data at the end of the month. Therefore, this project aims to propose a solution to solve the problem of having excess cellular data. This project plans to develop a mobile hotspot tethering system with a payment system using NFT to help people deal with their excess cellular data. Users will be offered the ability to share their excess Internet access and be rewarded during the process. This mobile application will also implement blockchain and smart contracts in the payment system, providing users with a better way to interact and facilitating the token transaction process.

# TABLE OF CONTENTS

<b>TITLE PAGE</b>	<b>i</b>
<b>REPORT STATUS DECLARATION FORM</b>	<b>ii</b>
<b>FYP THESIS SUBMISSION FORM</b>	<b>iii</b>
<b>DECLARATION OF ORIGINALITY</b>	<b>iv</b>
<b>ACKNOWLEDGEMENTS</b>	<b>v</b>
<b>ABSTRACT</b>	<b>vi</b>
<b>TABLE OF CONTENTS</b>	<b>vii</b>
<b>LIST OF FIGURES</b>	<b>x</b>
<b>LIST OF TABLES</b>	<b>xiii</b>
<b>LIST OF ABBREVIATIONS</b>	<b>xiv</b>
<b>CHAPTER 1 INTRODUCTION</b>	<b>1</b>
1.1 Problem Statement and Motivation	1
1.2 Project Scope and Direction	2
1.3 Project Objectives	2
1.4 Contributions	3
1.5 Background Information	4
1.6 Report Organization	5
<b>CHAPTER 2 LITERATURE REVIEW</b>	<b>7</b>
2.1 Previous Mobile Application on Tethering	7
2.1.1 PdaNet+	7
2.1.2 EasyTether Lite	9
2.1.3 Althea Mobile	10
2.1.4 Mobile Hotspot	12
2.1.5 Simplify	14
2.2 Summary of reviewed mobile application	17
2.3 Proposed solution for this project	19

<b>CHAPTER 3 SYSTEM METHODOLOGY/APPROACH</b>	<b>20</b>
3.1 Project Methodology	20
3.1.1 Overview of Agile SDLC Methodology	20
3.1.2 Breakdown of the phases in Agile SDLC	22
3.2 System Design Diagram	23
3.2.1 System Architecture Diagram	23
3.2.2 Use Case Diagram	24
3.2.3 Activity Diagram	26
 <b>CHAPTER 4 SYSTEM DESIGN</b>	 <b>34</b>
4.1 System Block Diagram	34
4.2 System Components Specifications	35
4.2.1 Connection Module	35
4.2.2 Nearby Hotspot Module	38
4.2.3 Payment Module	39
 <b>CHAPTER 5 SYSTEM IMPLEMENTATION</b>	 <b>41</b>
5.1 Hardware Setup	41
5.2 Software Setup	42
5.3 User Requirements	43
5.4 Setting and Configuration	43
5.5 System Operation	50
5.5.1 Login Functionality	50
5.5.1 Advertising and Discovering of Hotspot	54
5.5.1 Nearby Hotspot Finder	60
5.5.1 Payment Module	61
5.5.1 Settings	64
5.6 Implementation Issues and Challenges	66
5.7 Concluding Remark	68

<b>CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION</b>	<b>69</b>
6.1 System Testing and Performance Metrics	69
6.2 Testing Setup and Result	69
6.3 Project Challenges	76
6.4 Objectives Evaluation	77
<b>CHAPTER 7 CONCLUSION AND RECOMMENDATION</b>	<b>79</b>
7.1 Conclusion	79
7.2 Recommendation	80
<b>REFERENCES</b>	<b>82</b>
<b>APPENDIX</b>	<b>84</b>
<b>WEEKLY LOG</b>	<b>84</b>
<b>POSTER</b>	<b>90</b>
<b>PLAGIARISM CHECK RESULT</b>	<b>91</b>
<b>FYP2 CHECKLIST</b>	<b>93</b>

## LIST OF FIGURES

<b>Figure Number</b>	<b>Title</b>	<b>Page</b>
Figure 1.1	Mobile tethering through smartphones	4
Figure 1.2	Overview of smart contract	5
Figure 2.1	PdaNet+ mobile application UI	8
Figure 2.2	EasyTether Lite Homepage UI	9
Figure 2.3	Althea Mobile Homepage UI	11
Figure 2.4	Mobile Hotspot's Homepage UI	12
Figure 2.5	QR Code generator for hotspot	13
Figure 2.6	Simplify Homepage UI	14
Figure 2.7	Simplify nearby available hotspot map	15
Figure 2.8	Simplify in-app currency (GIGA) page	16
Figure 2.9	Simplify achievement page	16
Figure 3.1	6 phases of Agile SDLC	20
Figure 3.2.1	System Architecture Diagram of the application	23
Figure 3.2.2	Use Case Diagram of the application	25
Figure 3.2.3	Activity diagram for use case "Login"	26
Figure 3.2.4	Activity diagram for use case "Advertise Hotspot"	27
Figure 3.2.5	Activity diagram for use case "Discover Hotspot"	28
Figure 3.2.6	Activity diagram for use case "Configure network settings"	29
Figure 3.2.7	Activity diagram for use case "Nearby Hotspot Finder":	30
Figure 3.2.8	Activity diagram for use case "Payment Module"	31
Figure 3.2.9	Activity diagram for use case "Purchase Voucher"	32
Figure 3.2.10	Activity diagram for use case "FAQ page"	32
Figure 3.2.11	Figure 3.2.11: Activity diagram for use case "Provide Feedback"	33
Figure 3.2.12	Activity diagram for use case "Sign Out"	33
Figure 4.1	System Block Diagram	34
Figure 4.2.1	Process of the Nearby Connections	35
Figure 4.2.2	Block diagram of advertise hotspot function	36

Figure 4.2.3	Block diagram of discover hotspot function	37
Figure 4.2.4	Block diagram of nearby hotspot finder function	39
Figure 4.2.5	Transfer function in the smart contract	40
Figure 4.2.6	Block diagram of payment module	40
Figure 5.4.1	Developer options of mobile device	44
Figure 5.4.2	Declaring permissions in the AndroidManifest.xml	44
Figure 5.4.3	List of the flutter dependencies used	45
Figure 5.4.4	Initialization of Firebase	46
Figure 5.4.5	Setting up Google Maps API with API key	47
Figure 5.4.6	Setting up the Infura API using Web3Client	48
Figure 5.4.7	Calling the smart contract in flutter app	48
Figure 5.4.8	Contents of the smart contract	49
Figure 5.5.1	Logo of “Share&Link” application.	50
Figure 5.5.1	Application Splash	50
Figure 5.5.2	Onboarding screens	50
Figure 5.5.3	Application’s login page	51
Figure 5.5.4	Application’s reset password page	52
Figure 5.5.5	Email with a reset password link	52
Figure 5.5.6	Application’s Sign Up page	53
Figure 5.5.7	Example of stored user login credentials	53
Figure 5.5.8	Application connection page	54
Figure 5.5.9	Advertising hotspot	55
Figure 5.5.10	Discovering hotspot	55
Figure 5.5.11	List of nearby available advertiser	55
Figure 5.5.12	On connection initiation between discoverer and advertiser	56
Figure 5.5.13	Process of sending the hotspot duration	57
Figure 5.5.14	Incoming data page	58
Figure 5.5.15	Send hotspot credentials page	58
Figure 5.5.16	Hotspot information page	58
Figure 5.5.17	Countdown timer for the hotspot duration	59
Figure 5.5.18	Nearby Hotspot Finder Map	60
Figure 5.5.19	Tooltip for nearby available hotspot	60
Figure 5.5.20	Navigate to the hotspot location using Google Maps	61

Figure 5.5.21	Application's wallet page	62
Figure 5.5.22	Blocks creation on the smart contracts in the blockchain	62
Figure 5.5.23	Voucher Catalogue Page	63
Figure 5.5.24	Example of the order information in database	63
Figure 5.5.25	Application's settings page	64
Figure 5.5.26	Application's Feedback page	65
Figure 5.5.27	Application's FAQ page	65

## LIST OF TABLES

<b>Table Number</b>	<b>Title</b>	<b>Page</b>
Table 2.1	Summary of reviewed mobile application	17
Table 3.1	Breakdown of the phases in Agile SDLC	18
Table 5.1.1	Specifications of computer	41
Table 5.1.2	Specifications of smartphone 1	41
Table 5.1.3	Specifications of smartphone 2	42
Table 5.2.1	List of software used	42
Table 5.3.1	User connectivity requirement	43
Table 6.2.1	Test results for Login Page	70
Table 6.2.2	Test results for Forgot password page	70
Table 6.2.3	Test results for Sign up page	71
Table 6.2.4	Test results for Main page	71
Table 6.2.5	Test results for Connection module page	72
Table 6.2.6	Test results for Advertise Hotspot Functionality	72
Table 6.2.7	Test results for Discover Hotspot Functionality	73
Table 6.2.8	Test results for Nearby Hotspot Finder Map	74
Table 6.2.9	Test results for Payment Module	75
Table 6.2.10	Test results for Settings Page	75



## LIST OF ABBREVIATIONS

<i>UI</i>	User Interface
<i>API</i>	Application Programming Interface
<i>QR</i>	Quick Response
<i>USB</i>	Universal Serial Bus
<i>SSID</i>	Service Set Identifier
<i>WPA2</i>	Wi-Fi Protected Access II
<i>ISP</i>	Internet Service Provider
<i>VPN</i>	Virtual Private Network
<i>SDLC</i>	Software Development Life Cycle

# Chapter 1 Introduction

This chapter will cover the problem statement, project scope and objectives, and also the contributions and background information of this project.

## 1.1 Problem Statement and Motivation

The Internet has become an essential element in daily human life. According to the statistics provided by Statista Research Department [1], there were more than five billion active Internet users globally as of April 2022, accounting for 63.1 per cent of the global population. Among the numbers of the active Internet user, 4.32 billion of them are actively accessing the web via a mobile device. Moreover, Josh [2] stated that 62.06% of website traffic is generated mainly from mobile devices. The statistics above emphasise the importance and significance of Internet access from a mobile device.

Followed by the rise of the Internet and mobile devices, mobile cellular service has reached a new height in their records. The world's total mobile cellular subscriptions had increased by 40% within 10 years (2010 to 2020), which is from 76 subscriptions per 100 people in 2010 to 106 subscriptions per 100 people in 2020 [3]. However, an excessive amount of mobile cellular subscriptions has caused the user to have an excess mobile data tariff remaining at the end of every month. Ernest [4] stated that an average user wastes 3.4 GB of mobile data every month. Collectively, United Kingdom mobile customers pay for 143 million gigabytes of data each month that they do not use. As a result, mobile customers are wasting their money paying for extra mobile data they do not use. Therefore, this project claims that it is important to utilise the excess mobile data sufficiently by reselling it to others who need it.

Moreover, people who do not have a mobile cellular subscription or have a weaker cell phone signal in certain areas could have difficulties accessing the Internet. Hence, this project proposes a highly available and more accessible way to connect to the Internet by using others' mobile hotspots with certain charges.

### 1.2 Project Scope and Direction

This project aims to deliver a hotspot-tethering mobile application integrated with a payment system. Besides, this project intends to provide a platform for users to share and sell their excess mobile data through hotspot tethering. Users could recover some expenses on the mobile cellular bill they spent. On the other hand, this project aims to deliver a network with high coverage and availability. It provides easier access to the Internet as every user could be the “moving hotspot”. Also, an In-app virtual currency will be implemented to handle the transactions.

### 1.3 Project Objectives

The objectives of this project are:

- i. **To develop a hotspot-tethering mobile application that provides a platform for mobile data sharing using Flutter**

A mobile application with multiple features will be developed using Flutter with Dart programming languages. Primarily, this mobile application will enable users to sell excess mobile data by creating a mobile hotspot that others can connect to. Alternatively, users who require Internet access can connect to the mobile application’s provided hotspot.

- ii. **To implement smart contracts in the payment system in the application using Solidity**

A payment system with smart contracts will be implemented and integrated with the mobile application for hotspot tethering. In the application, payment is automated using smart contracts. Once a user has successfully provided Internet access via mobile hotspot tethering, the other party will automate payment via the predefined smart contract.

### **iii. To facilitates the process of finding Internet access through a nearby hotspot finder using Google Maps API**

The hotspot-tethering application will include a nearby hotspot finder to make finding a hotspot easier. Users who initiate hotspot tethering will have their coordinates collected and stored in a database. The hotspot's location will be marked and displayed on the application's Google Maps widget using the Google Maps API. Users can identify nearby hotspots by examining the map's markers.

## **1.4 Contributions**

The majority of individuals have excess mobile data at the end of the month. Therefore, it is essential to develop a mobile application for hotspot tethering that provides a platform for users to sell excess mobile data and locate an Internet access point when needed.

This project proposes a mobile application for students and faculty at the University of Tunku Abdul Rahman that utilises hotspot tethering. This application enables them to sell their excess mobile data to others in order to recover a portion of their cell phone bill costs. Alternatively, a user who has exhausted their mobile data plan could easily connect to the hotspot to access the Internet. In addition, this project aims to expand network availability coverage within the university campus. This hotspot-tethering mobile application would allow users with weak cell phone signal reception to have access to the Internet.

## **1.5 Background Information**

### **1.5.1 Hotspot tethering**

Mihai et al. [5] describe that the tethering process involves forwarding traffic by bridging the 3G or 4G interface with Wi-Fi, Bluetooth, or USB from one network interface to the other. Most devices are usually equipped with hardware capabilities to allow hotspot tethering in the current meta. This methodology also states that the

Internet Protocol (IP) gateway solution is currently the most abundantly used tethering mechanism. IP gateway solution uses smartphones to act as the IP router and gateway for the local area network, which will forward IP packets through the gateway (Figure 1.1). Besides that, Network Address Translation (NAT) is also implemented in the IP gateway to translate. With this implementation, smartphones can act as IP routers with NAT, thus forwarding the packets through a General Packet Radio Service (GPRS) tunnel. Implementations such as Wi-Fi direct and ad hoc have also been discussed in the project. Technically, ad hoc has been supported by many devices, but Wi-Fi direct is not. Wi-Fi direct is an implementation that will automatically configure one of the devices as a soft application, bringing significant security improvements, more straightforward configurations, and higher performance.

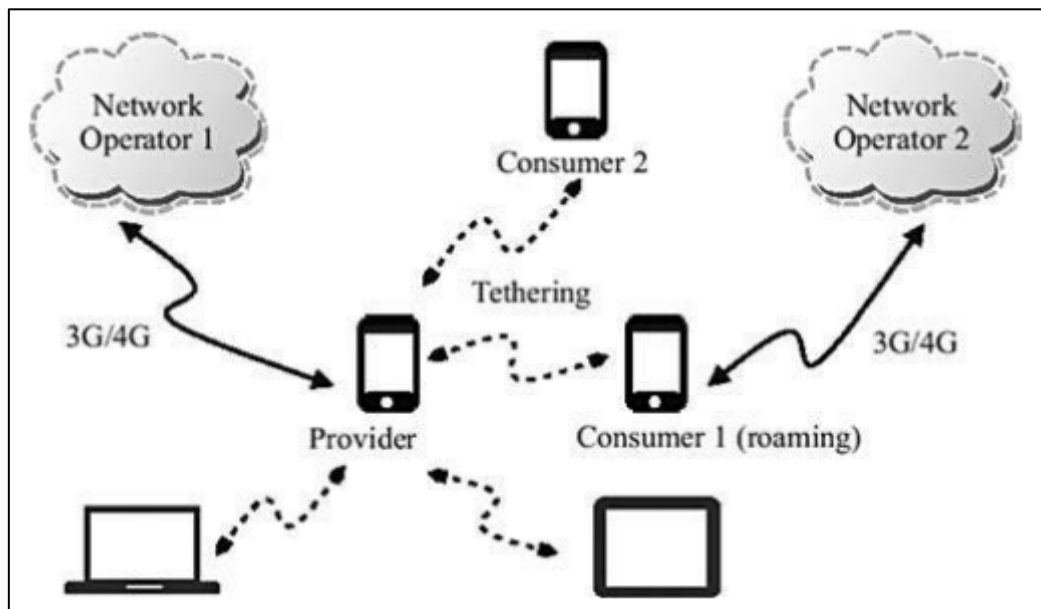


Figure 1.1: Mobile tethering through smartphones

### 1.5.2 Smart contracts

According to [6] and [7], smart contracts are automated, self-executing contracts with the terms of an agreement between a seller and a buyer. The agreement is encoded in lines of code and distributed across a decentralised blockchain network. For instance, a smart contract will automatically pay the service provider upon completion of the work. As smart contracts are more efficient, transparent, and secure in payment systems [8],

they should be implemented in this project. The overview of smart contracts is shown below (Figure 1.2).

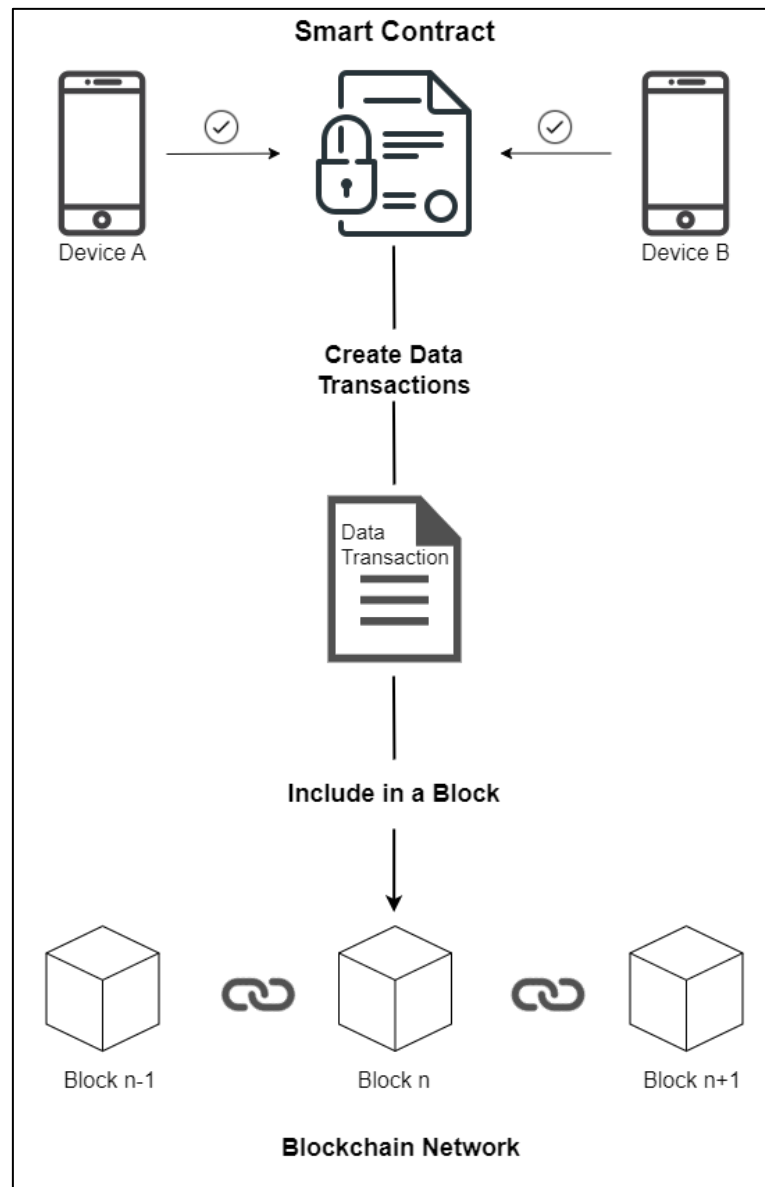


Figure 1.2: Overview of smart contract

### 1.6 Report Organization

This report is organized into seven chapters, providing a comprehensive overview of the development and evaluation of the mobile hotspot tethering application. The chapters are as follows.

## CHAPTER 1 INTRODUCTION

Chapter 1 presents the problem statement, motivation, objectives, project scope and direction, contributions, and report organization. Then, chapter 2 reviews previous works on mobile application on tethering, critically analyses the existing solutions, and proposes improvements. Chapter 3 discusses the system methodology and the system design diagram. The system design diagrams include diagram such as use case and activity diagrams. Next, chapter 4 provides an overview of the system and presents the system block diagram. Chapter 5 details the hardware setup, software setup, settings and configurations, system operation, implementation issues and challenges, and concluding remarks. Chapter 6 focuses on system testing and performance metrics, project challenges, and objectives evaluation. Lastly, Chapter 7 presents the conclusion and provides recommendations for future improvements and enhancements to the mobile tethering and sharing application.

## Chapter 2 Literature Review

This section will review previous mobile applications for hotspot tethering. Each application will be evaluated based on its respective strengths and weaknesses. Then, a summary table of this review will be presented. Finally, a proposed resolution will be discussed.

### 2.1 Previous mobile application on tethering

#### 2.1.1 PdaNet+

PdaNet+ [9] is a mobile application focused on hotspot tethering that can be used for tethering between devices. This application was created by the June Fabrics Technology team. The application functions by sharing an Android mobile device's Internet connection with other devices, such as computers, tablets, and other mobile devices. There are several tethering options available to the user, including USB tethering, Bluetooth tethering, and Wi-Fi Direct Hotspot tethering. Tethering via USB option permits an Android mobile device and a computer to share an Internet connection via a USB cable. Meanwhile, Tethering via Bluetooth enables Bluetooth tethering, but this tethering solution is not optimal due to its slow transfer speed. Notably, the Wi-Fi Direct Hotspot feature enables Wi-Fi Direct tethering between an Android mobile device and a computer, tablet, or another mobile device. This Wi-Fi Direct Hotspot option creates a Wi-Fi direct connection between devices so that they can share Internet access. The procedure involves reconfiguring the Group Owner tethering device's Wi-Fi chip into access point mode [10]. After that, the tethered device will function as an access point for other devices to connect and gain Internet access. A proxy and a standard WPA2 protocol are involved in the connection. In addition, this application enables users to configure hotspot information to facilitate connection. This application's user interface is provided below (see Figure 2.1).



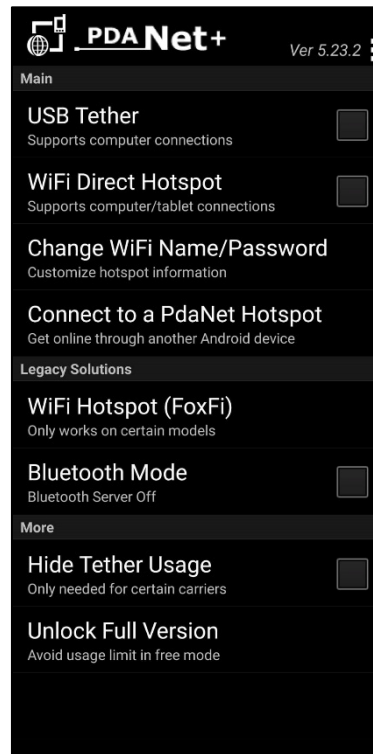


Figure 2.1: PdaNet+ mobile application UI

### Strengths

**Multiple tethering options** are provided, allowing users to select their preferred tethering method. In addition, USB Tethering offers faster Internet speed [11], is more efficient due to less battery drain, and enables long-term device connections. Besides, users could **configure their preferred hotspot password and band rate** (2.4 GHz or 5 GHz) for the provided hotspot to enhance security and personalisation.

### Weaknesses

However, **only Wi-Fi tethering** is supported by this application. This means that only devices that are connected to Wi-Fi can tether to the other device. **Tethering of mobile data is prohibited.** This application did not allow users to share their mobile data through hotspot tethering. Besides that, this application **only supports one-to-one connection**, which means only one device can be connected to the tethered connection only. In addition, this application restricts the usage and Internet speed while tethering, requiring users to pay to remove the restrictions.

### 2.1.2 EasyTether Lite

The mobile application EasyTether Lite [12] enables tethering between two devices. Mobile Stream was responsible for creating this application. This application primarily operates on Android devices. This application offers two essential features: USB Tethering and Bluetooth Tethering. A user could tether his Android device's internet connection to a computer using the Tethering via USB mode. Afterward, the computer will have Internet access via the tethering of the. Additionally, this USB Tethering is compatible with numerous operating systems, including Windows, Mac OS, Android, and Linux. Under the Tethering via Bluetooth mode, an Android user could tether his device to other Bluetooth-capable devices. Each tethering method includes a setup wizard to assist the user during the tethering procedure. The figure below (Figure 2.2) shows the UI of EasyTether Lite.

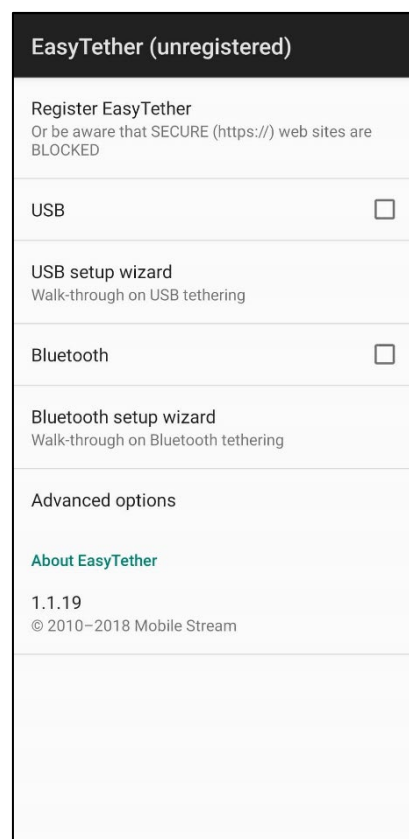


Figure 2.2: EasyTether Lite Homepage UI

### **Strengths**

There are two tethering options available, allowing users to choose their preferred method. Additionally, USB Tethering provides a more **stable and rapid connection** between devices. It also **supports long-term connection** between devices during the tethering process.

### **Weaknesses**

This application **does not support mobile data hotspot tethering**, only tethering through Wi-Fi connection. Besides that, this application is uneasy to use as it requires **additional configuration**, such as USB debugging prior to the use of the Tethering via USB feature. In addition, this application is a lite version, where the tethered connection will **block HTTPS connections on port 443, instant messaging, and online gaming**. As a result, it prevents users from accessing certain websites on the Internet. In order to remove the restriction, users must pay to unlock the full potential and functionality of this application.

### **2.1.3 Althea Mobile**

This application named Althea Mobile [13] allows users to purchase bandwidth via a Wi-Fi hotspot from an Althea.net router. This application was developed by the United States-based ISP organisation Althea. Althea is a decentralised ISP, unlike conventional ISPs, where each user owns their own router and connects to an open network of devices that buy and sell bandwidth. This application uses a WireGuard VPN to secure users' Internet connections. The mobile application enables users to connect to the Althea Internet via the Althea ISP-provided Wi-Fi network. Certain fees will be assessed based on the user's consumption. Internet usage will also be displayed so that users can see how much Internet they have consumed. In addition, the user's account balance is displayed. Users can deposit and withdraw funds from their balances. (Figure 2.3)

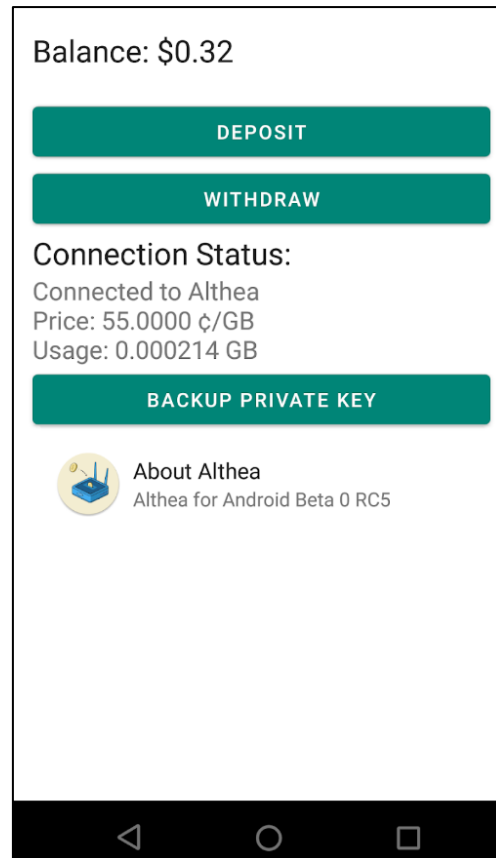


Figure 2.3: Althea Mobile Homepage UI

### **Strengths**

This application **supports deposit and withdrawal functions** for Internet access purchases and sales. Where users could deposit funds into the application's balance to purchase Internet access, a user may also withdraw their funds if they earn money by selling Internet access to others. In addition, VPN is also used on the connection established between devices to **secure Internet connection**.

### **Weaknesses**

This application is **compatible only with the Althea ISP router**. Only the Althea router was capable of Internet access sharing. In addition, this application can only connect to the Althea router in order to gain Internet access. Other routers are unavailable for such operations.

### 2.1.4 Mobile Hotspot

Mobile Hotspot [14] is a mobile application that gives users customisation and configuration on their hotspot tethering system. This application allows users to toggle hotspot tethering on/off through a switch. In addition, it provides several control options for mobile hotspot tethering (Figure 2.4). The functions provided are a timer function, network speed test, data limiter, and battery limiter.

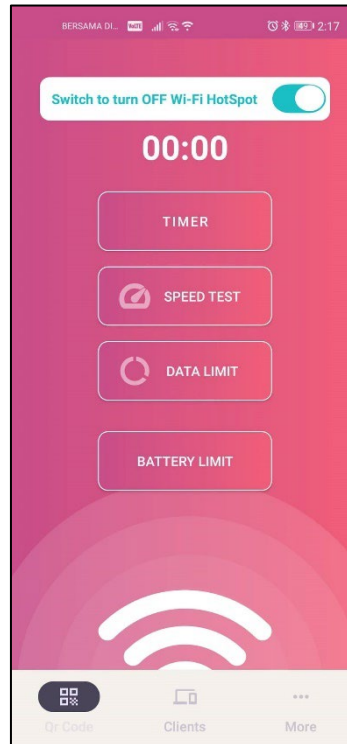


Figure 2.4: Mobile Hotspot's Homepage UI

First of all, the timer function gives the user to define a preferred timer, where the hotspot tethering will automatically turn off as the timer expires. Next, a network speed test function is provided for a user to test its network speed. A user could also define the data limit to be used by the connected device. As the limit reaches, the hotspot tethering will be turned off, and a notification will pop up to notify the user. A user could also set the battery limit to prevent the hotspot from tethering when the battery level reaches a certain threshold. Besides that, a list of connected devices will be displayed to the user. A user could view the list of devices connected to the hotspot. Notably, this application implements a QR code generator function to generate the QR code for the hotspot (Figure 2.5). In order to generate the hotspot QR code, the user will be prompted to enter the hotspot's SSID and password.

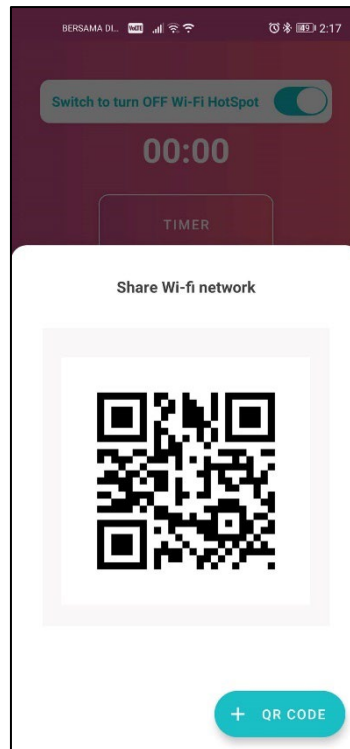


Figure 2.5: QR Code generator for hotspot

### Strengths

The **QR Code generator** for the hotspot facilitates the connection of other devices to the hotspot. Besides, a number of features are provided so that the user has complete control over their own hotspot tethering configuration. In addition, it provides an **in-app network speed test** for users to check their own network speed instantly.

### Weaknesses

This application lets the user **share mobile hotspot tethering without any charges applied**. No fees will be assessed to devices that connect to the hotspot. In other words, this application does not enhance the hotspot tethering system with integration with payment system; it merely provides additional configuration options for the hotspot.

### 2.1.5 Simplify

Simplify [15] is a mobile application that allows users to sell and find Internet access through hotspot tethering, which is integrated with an In-app currency payment system. This mobile application was developed by Simplify Networks based in Kuala Lumpur, Malaysia. The objectives of this mobile application are to provide democratised mobile connectivity and bring Internet-on-demand to people globally. Primarily, a user could sell or find the Internet through the Simplify application (Figure 2.6). The Internet provider name, Internet speed, and signal strength are also indicated.

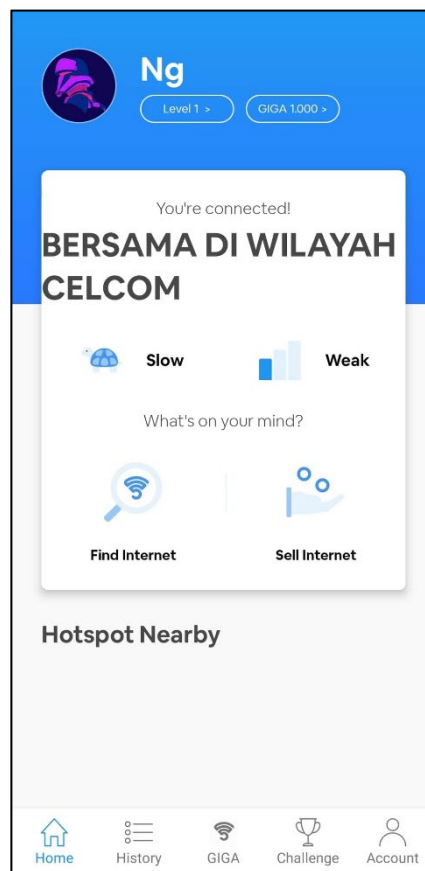


Figure 2.6: Simplify Homepage UI

If a user wishes to sell the Internet, the application will prompt the user to turn on the mobile device hotspot with a specified SSID and password. After the user turns on the hotspot, the user is ready to sell his mobile data through hotspot tethering. Any devices connected to the hotspot will be displayed in a list. Users can get earnings by sharing mobile hotspots with other devices. If a user wishes to find Internet access

## CHAPTER 2 LITERATURE REVIEW

through Simplify application, a map using MapBox API will display all nearby available hotspots to the user (Figure 2.7).

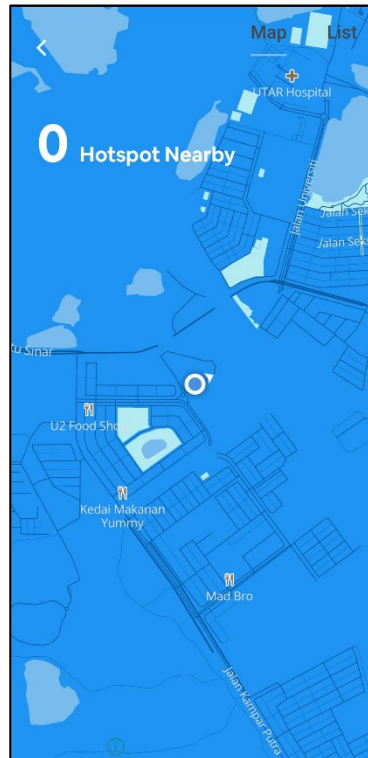


Figure 2.7: Simplify nearby available hotspot map

The available hotspots will be displayed on a map or in a list, and the user can select the desired hotspot. Upon connecting to the hotspot, fees will be incurred. In addition, Internet usage and earnings from selling Internet access are displayed on the History page. The user is able to determine how much mobile data has been consumed and how much they have earned by selling their mobile data.

Notably, this application implemented an in-app currency to manage the transaction of selling and buying Internet access. A currency named GIGA is introduced, with the rate of 1 GB of data per 1 GIGA. Users can earn GIGA by selling their mobile Internet. Alternatively, a user must pay GIGA to the hotspot provider if they used the other's mobile hotspot through this application. In addition, a user can check the current GIGA balance. A user could also add funds to the app to purchase Internet access from the app (Figure 2.8).



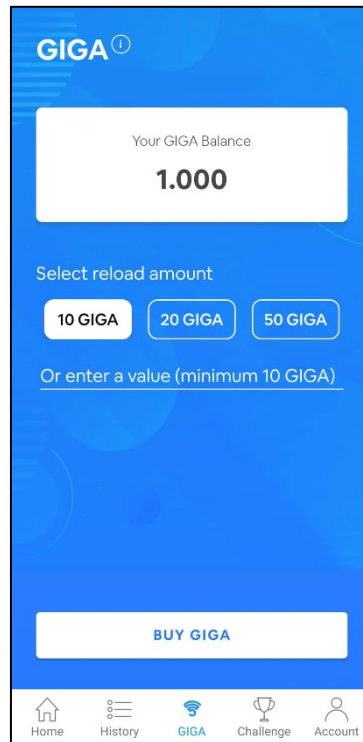


Figure 2.8: Simplify in-app currency (GIGA) page

Additionally, an achievement system is also being implemented in this application (Figure 2.9). This system will reward the user with unique badges based on the time spent and user engagement towards the application.

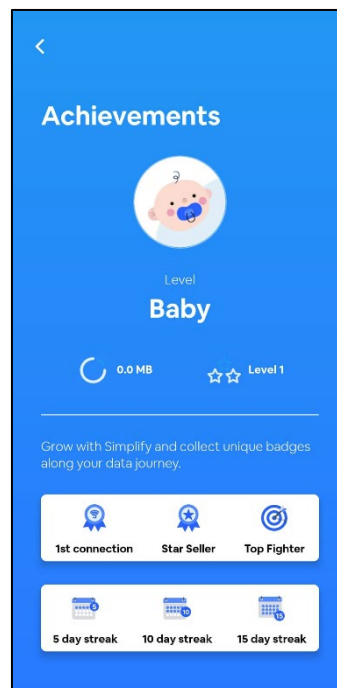


Figure 2.9: Simplify achievement page

### **Strengths**

The implementation of an in-app currency payment system within the application **allows users to gain earnings** when sharing mobile data via hotspot tethering. This implementation enables users to sell unused mobile data and recoup a portion of their mobile cellular subscription costs. Moreover, the **implementation of a geolocation API** that displays nearby available hotspots benefits the application user because the user can quickly locate the physical location of the hotspot. Additionally, the **achievement system** increases user engagement with this application, resulting in users with greater levels of engagement and more time spent on this application.

### **Weaknesses**

The in-app currency payment system **does not support the ability to withdraw in-app funds**. This situation indicates that the funds added to the application will remain within the application indefinitely. For instance, if a user earns 5 GIGA by selling its mobile data to others, the user cannot convert the 5 GIGA to another currency; the earnings of GIGA can only be used within the application to purchase Internet access from other users. Consequently, there is no way to withdraw earnings from this application to purchase other items in the real world. In addition, there is an exploit where a user could terminate the application after successfully connecting to a hotspot, resulting in no charges will be applied to the connection.

## **2.2 Summary of reviewed mobile application**

The table below shows the summary of the reviewed mobile application:

<b>Existing Mobile Application</b>	<b>Capabilities to tether mobile data</b>	<b>Strength</b>	<b>Weakness</b>
------------------------------------	---	-----------------	-----------------

## CHAPTER 2 LITERATURE REVIEW

PdaNet+	✗	<ul style="list-style-type: none"> <li>- Multiple tethering options provided</li> <li>- Custom configuration on hotspot password and band rate</li> </ul>	<ul style="list-style-type: none"> <li>- Supports Wi-Fi tethering only</li> <li>- Mobile data tethering is prohibited</li> </ul>
EasyTether Lite	✗	<ul style="list-style-type: none"> <li>- Stable and rapid connection</li> <li>- Supports long-term connection</li> </ul>	<ul style="list-style-type: none"> <li>- Supports Wi-Fi tethering only</li> <li>- Mobile data tethering is prohibited</li> <li>- Blocks certain connections</li> </ul>
Althea Mobile	✗	<ul style="list-style-type: none"> <li>- Supports deposit and withdrawal functions</li> <li>- Used VPN for secure Internet connection</li> </ul>	<ul style="list-style-type: none"> <li>- Only compatible with the defined ISP router</li> </ul>
Mobile Hotspot	✓	<ul style="list-style-type: none"> <li>- In-app hotspot's QR code generator</li> <li>- Facilitates hotspot connection</li> <li>- In-app network speed test</li> </ul>	<ul style="list-style-type: none"> <li>- Mobile hotspot tethering that gives other Internet access without charges.</li> </ul>
Simplify	✓	<ul style="list-style-type: none"> <li>- Allows users to gain earnings through selling excess Internet</li> <li>- Maps that display nearby hotspot</li> <li>- Achievement system</li> </ul>	<ul style="list-style-type: none"> <li>- Does not support withdrawal of funds from the app</li> </ul>

Table 2.1: Summary of reviewed mobile application

### **2.3 Proposed solution for this project**

This project aims to propose a solution that could help implement a mobile hotspot tethering application that serves mainly the student and faculty of the University of Tunku Abdul Rahman. The application will be developed with multiple features. Specifically, the discover hotspot and advertise hotspot functions will be implemented as the primary functions. Through hotspot tethering, the advertise hotspot will allow users to share and provide their excess mobile data. Alternatively, the discover hotspot feature will allow the user to locate and connect to the hotspot internet access within the application. A map will be displayed with markers indicating nearby hotspots to assist users in locating a hotspot. The app will integrate with blockchain and also in-app tokens to manage the transactions after the hotspot sharing. Most importantly, it will be possible to withdraw earnings. A wallet token will be defined to facilitate smart contract transactions between parties. In addition, a clean and simple UI in the application will provide users with a straightforward, intuitive, and aesthetically pleasing mobile application experience.

## Chapter 3 System Methodology/Approach

### 3.1 Project Methodology

#### 3.1.1 Overview of Agile SDLC Methodology

This project adopts Agile SDLC in building the mobile hotspot tethering application. According to the research of Anitha et al. [16], Agile SDLC is the optimal methodology for developing mobile applications. This is due to the nature of Agile SDLC having a combination of iterative and incremental approaches in which the entire SDLC is divided into small iterations. Thus, Agile SDLC allows the project to adapt quickly to changes. Besides that, Agile SDLC has a cyclical process, where the work is regularly done in iterated circles. This allows numerous cycles of sprint and improvement on the final deliverable. The Agile SDLC is divided into 6 phases, which include: (I) Planning phase, (II) Analysis phase, (III) Design phase, (IV) Development phase, (V) Testing phase and lastly (V) Support and maintenance phase. Figure 3.1 demonstrates the 6 phases of the Agile SDLC process flow in a single sprint.

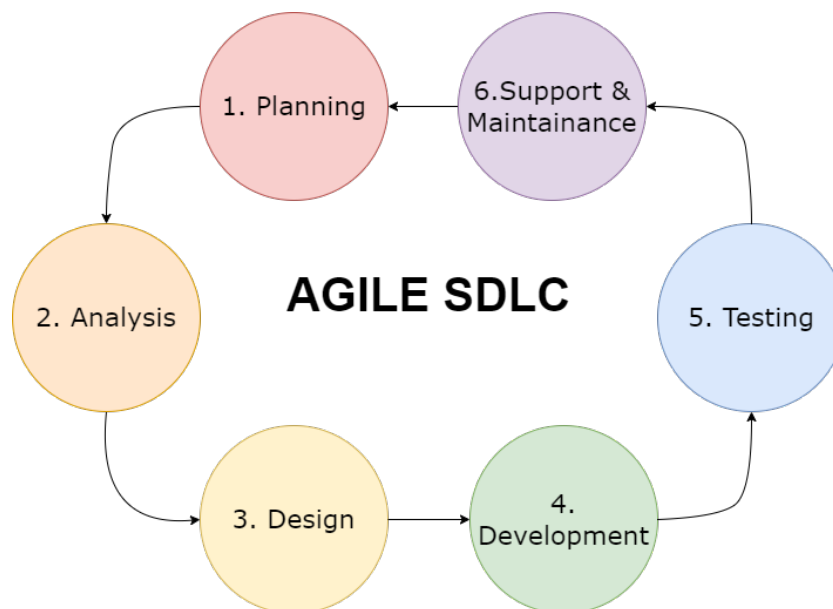


Figure 3.1: 6 phases of Agile SDLC

## CHAPTER 3 SYSTEM METHODOLOGY/APPROACH

The phases of Agile SDLC are breakdown as below:

### I. Planning phase

- Identify problems encountered by the users
- Define the project scope and project objectives
- Create a project timeline to monitor the project's progress

### II. Analysis phase

- Analyse user requirements on the application
- Determining the core functions to solve the problems

### III. Design phase

- Develop mobile application's system architecture
- Design application UI
- Identify the tools required in the development of the project

### IV. Development phase

- Develop the mobile application's function and features
- Determine the challenges and obstacles during the implementation

### V. Testing phase

- Perform alpha testing on the MVP developed
- Identify potential bugs and error outcome

### VI. Support and Maintenance

- Gather feedback from the alpha tester
- Analyse and document the feedback provided
- Start a new Agile SDLC sprint for further improvement

### 3.1.2 Breakdown of the phases in Agile SDLC

The table below provides an overview of the task accomplished in this project. Each of the development phases is broken down in detail with the corresponding tasks performed in each phase.

Development Phases	Task accomplished
Planning phase	<ul style="list-style-type: none"> <li>Identified the existing problems faced by users.</li> <li>Created a project timeline to plan the project's progress</li> </ul>
Analysis phase	<ul style="list-style-type: none"> <li>Reviewed 5 existing mobile hotspot tethering applications.</li> <li>Identified the problems that exist in the application.</li> <li>Analysed the strengths and weaknesses of each particular application.</li> <li>Defined the scope and objectives that this project aims to address.</li> </ul>
Design Phase	<ul style="list-style-type: none"> <li>Researched on the tools to be used for the proposed system development.</li> <li>Designed the User Interface (UI).</li> <li>Identified the main functions to be developed.</li> <li>Provided a brief detail on the flow of the system.</li> </ul>
Development phase	<ul style="list-style-type: none"> <li>Set up required hardware and software.</li> <li>Developed the essential modules of the project.</li> <li>Integrated the modules of the projects.</li> <li>Identified the challenges and issues while implementing the prototype.</li> <li>Make certain changes to the prototype to overcome the challenges.</li> </ul>
Testing phase	<ul style="list-style-type: none"> <li>Prepared a test case.</li> <li>Performed unit testing on the prototype.</li> <li>Recorded the outcomes of the testing process.</li> <li>Documented the results of the testing process.</li> </ul>
Support and Maintenance phase	<ul style="list-style-type: none"> <li>Prepared a questionnaire to obtain feedback from the users.</li> </ul>

Table 3.1: Breakdown of the phases in Agile SDLC

## 3.2 System Design Diagram

### 3.2.1 System Architecture Diagram

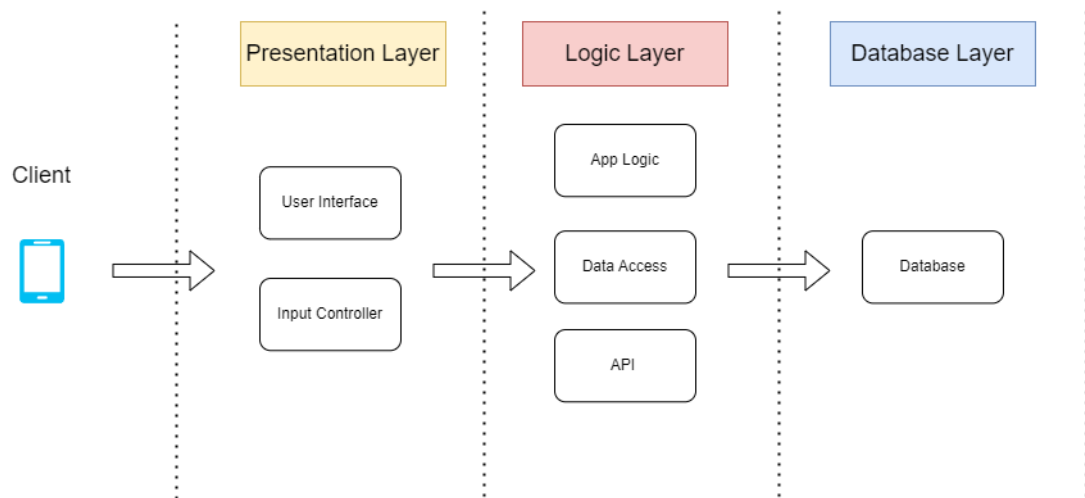


Figure 3.2.1: System Architecture Diagram of the application

The system architecture diagram for this project consists of 3 layers, which are the presentation layer, the logic layer, and also the database layer.

The presentation layer is responsible for the user interface and how the system interacts with users. It focuses on providing a visually appealing and user-friendly experience. Two key components of the presentation layer are the User Interface (UI) and also the input controller. The user interface (UI) refers to the visual elements and controls that facilitate user interaction with a system. It consists of icons, forms, menus, and other graphical user interface elements. The input controller is responsible for managing the communication between the user interface and the logic layer. It processes user inputs such as finger gestures and keyboard events and forwards them to the appropriate logic layer components for processing.

The logic layer contains the core functionalities and application logic of the system. It processes user inputs, performs the underlying logic of the functions, and manages the flow of data within the system. The logic layer consists of the following components: The application logic, API, as well as the Data access component. The application logic specifies the rules and operations necessary to complete particular duties or achieve desired outcomes. It implements algorithms, processes data, and

Bachelor of Information Technology (Honours) Communications and Networking  
Faculty of Information and Communication Technology (Kampar Campus), UTAR



controls the system's overall behaviour. While the Application Programming Interface (API) serves as an intermediary between software components, enabling them to communicate and exchange data. APIs permit integration with external systems and services, thereby expanding the application's capabilities. And also, the data access component is responsible for retrieving and manipulating stored data. It communicates with databases and other data sources in order to receive, write, update, and delete data as required by the system.

The database layer stores and manages the system's data. It provides a structured and organized way to store, retrieve, and manipulate data. The key component of the database layer is the database. The database serves as a central repository for structured data storage. It ensures data integrity, enforces data constraints, and facilitates efficient information querying and retrieval.

### **3.2.2 Use Case Diagram**

The use case diagram is shown in the figure below (Figure 3.3). The mobile hotspot tethering application's use case diagram will outline all the functions offered to users. The user will be able to log in or create an account to gain access to the functionalities provided by the mobile application. Then, user will be able to choose the desired actions to be done in the mobile application. All of the actions that can be done is included in the use case diagram.



Figure 3.2.2: Use Case Diagram of the application

### 3.2.3 Activity Diagram

#### Use Case: Login to the application

The user will be prompted to log in or create an account with their email and password when they launch the application. After logging in or creating an account successfully, the user will be directed to the home page. The user can use the forgot password button to reset their password by clicking on it. The application will send a link to their email to reset their password. This system uses Firebase authentication to verify the user's email and password during the login process, and if the user is unable to pass the verification, the system will display the relevant error message.

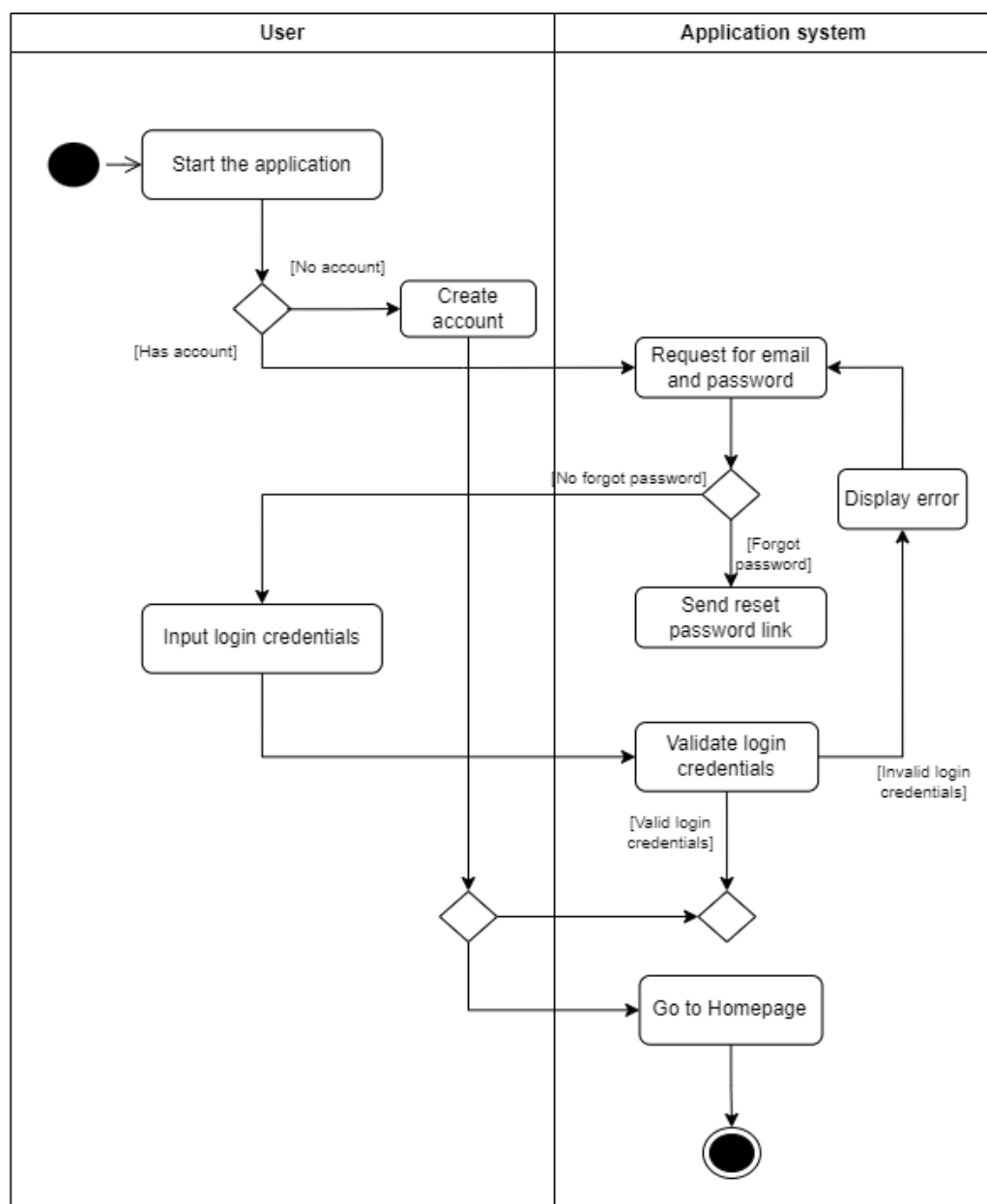


Figure 3.2.3: Activity diagram for use case “Login”

Use Case: Advertise Hotspot

When a user uses the Advertise Hotspot function, the user will be requested to turn on the advertise hotspot. After that, the nearby connection will be executed in a advertise mode and wait for an incoming connection. At the same time, the location data will be uploaded to the database. If there is an incoming connection, the user can choose to accept the connection and send wait for the client to send the hotspot duration they wanted to use. After receiving the duration, the users can then send the hotspot credentials to the client and a countdown timer will be started to track on the hotspot duration.

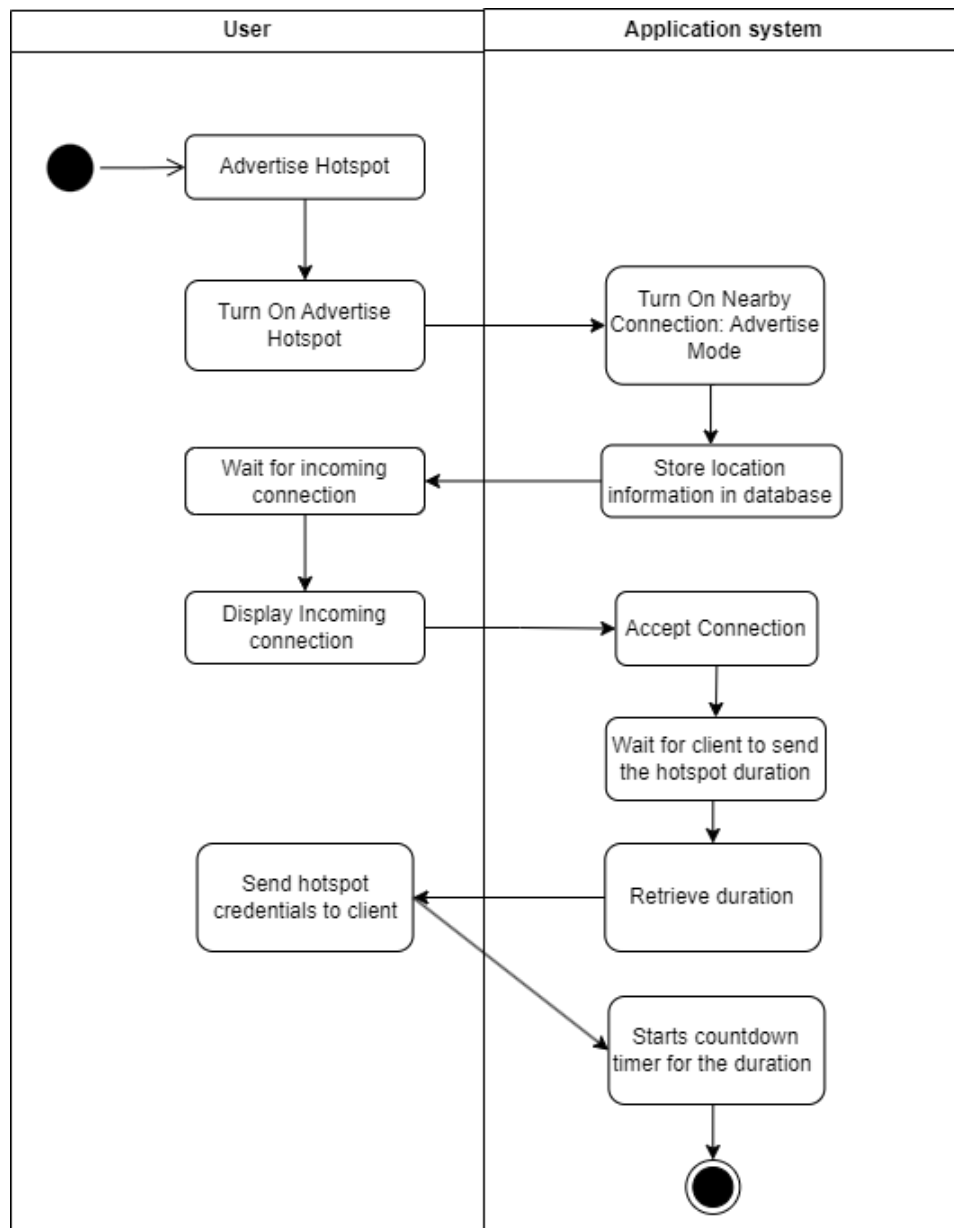


Figure 3.2.4: Activity diagram for use case “Advertise Hotspot”

Use Case: Discover Hotspot

When a user uses the Discover Hotspot function, the user will be requested to turn on the discover hotspot. After that, the nearby available advertiser will be retrieved and displayed. The user can select which hotspot they want to connect to. After that, it will initiate the connection with the hotspot advertiser and receive the hotspot credentials. After that the Hotspot QR Code will be displayed, and the user can scan the QR code to connect to the hotspot. After connecting to the hotspot, the duration used will be calculated as well.

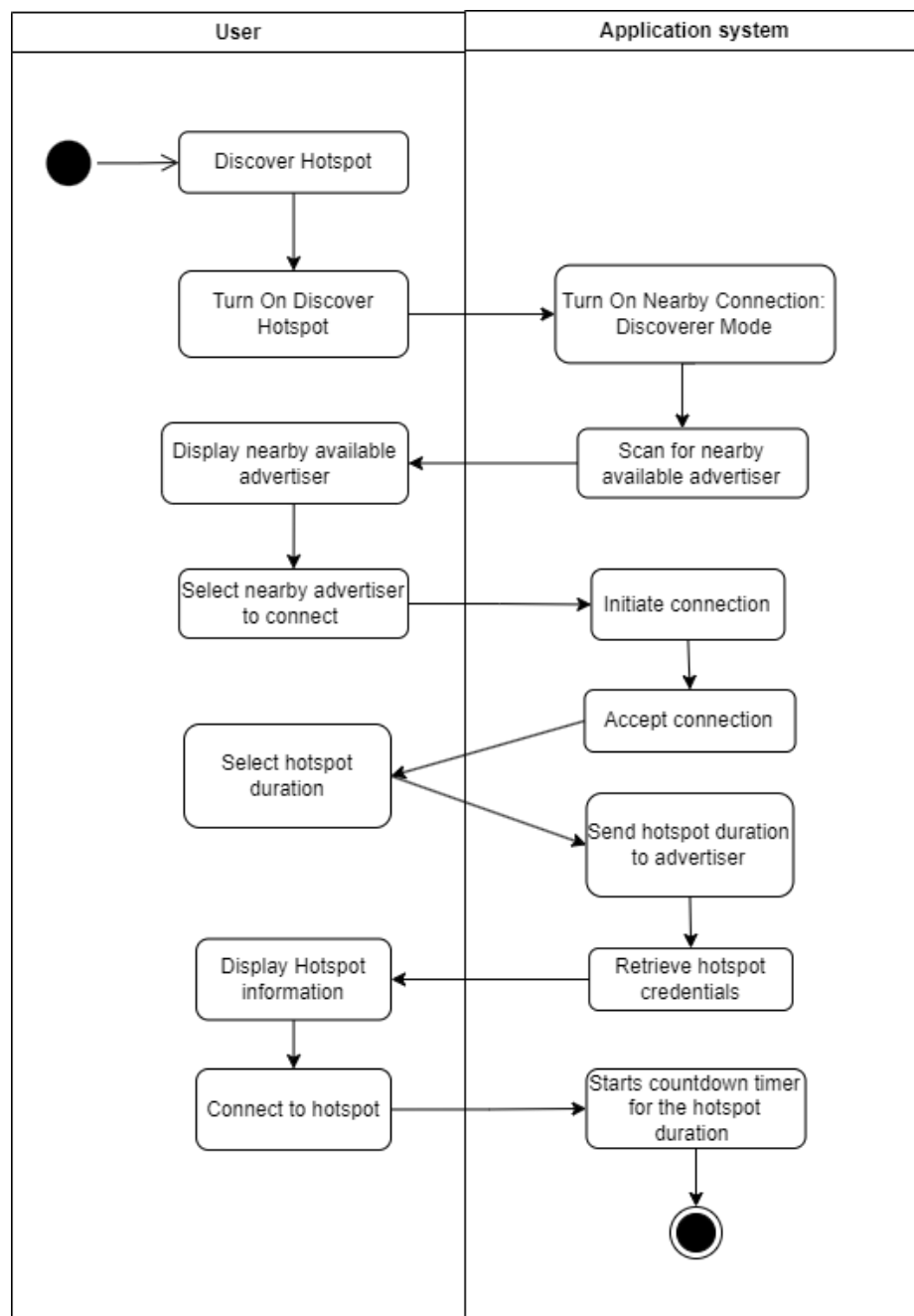


Figure 3.2.5: Activity diagram for use case “Discover Hotspot”

Use Case: Configure network settings

When a user wants to configure the network settings, the user will be prompted to select the settings to be configured. After that, the system will open up the android intent to the settings that are selected by the user.

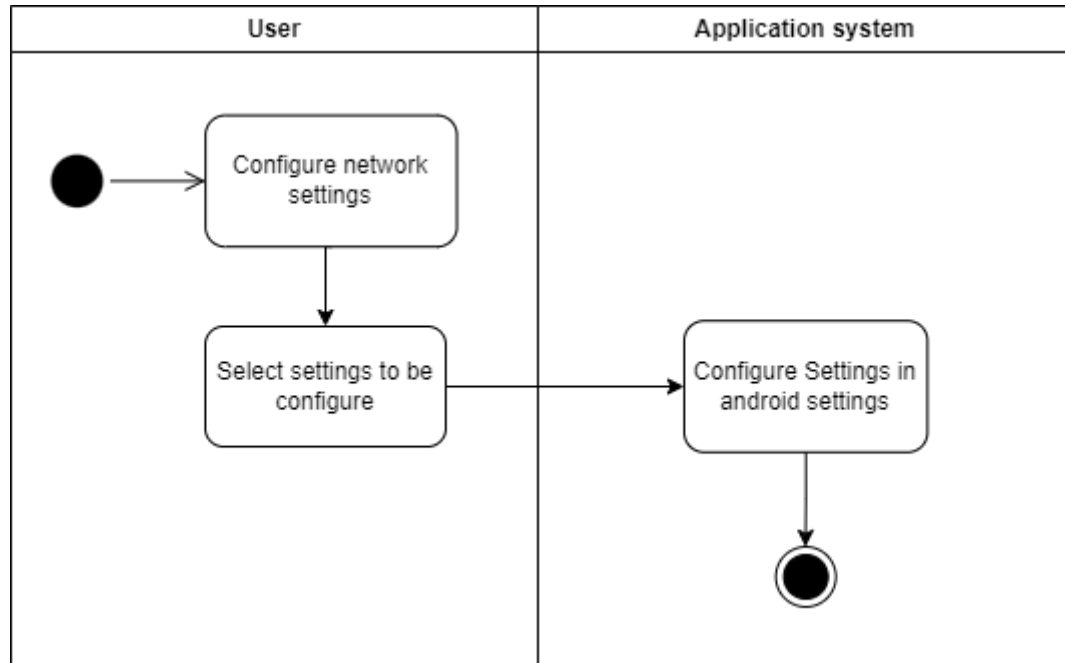


Figure 3.2.6: Activity diagram for use case “Configure network settings”

Use Case: Nearby Hotspot Finder

When a user opens the nearby hotspot finder page, firstly the application system will retrieve the location information from the database, after that it will filter out the nearby hotspot location based on the distance from the user and also the timestamp. Irrelevant locations will be filtered out. After that, the hotspot markers will be added to the map and then displayed to the user. Users can then select the desired hotspot and navigate to the hotspot location through Google Maps.

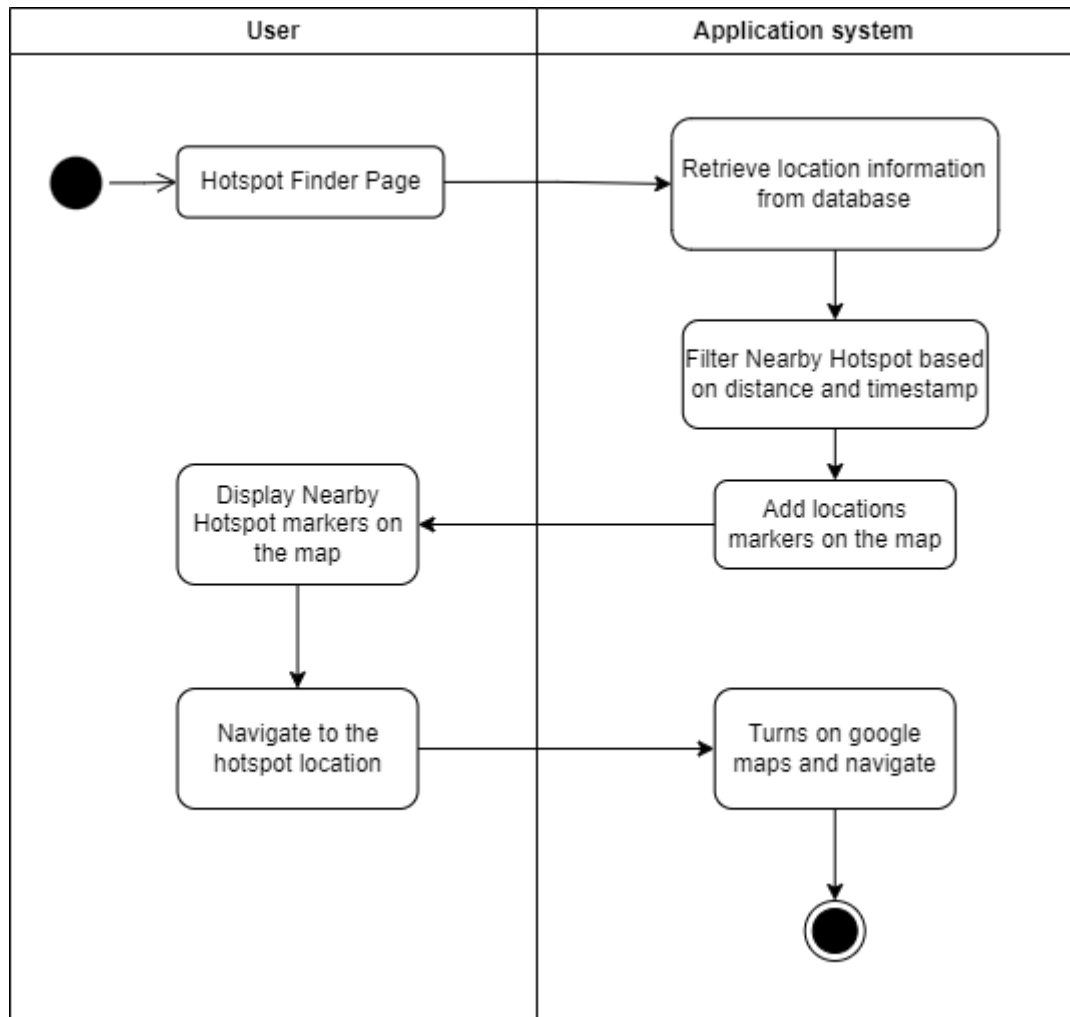


Figure 3.2.7: Activity diagram for use case “Nearby Hotspot Finder”:

Use Case: Payment Module

In the payment module, at first, the user can see the amount of tokens in the user’s wallet. Then users can select the token amount to be added, withdrawn, or transferred. After that, users can press the related buttons to perform the action. The system’s application system will then perform the payment operations then notify and update the status of the user.

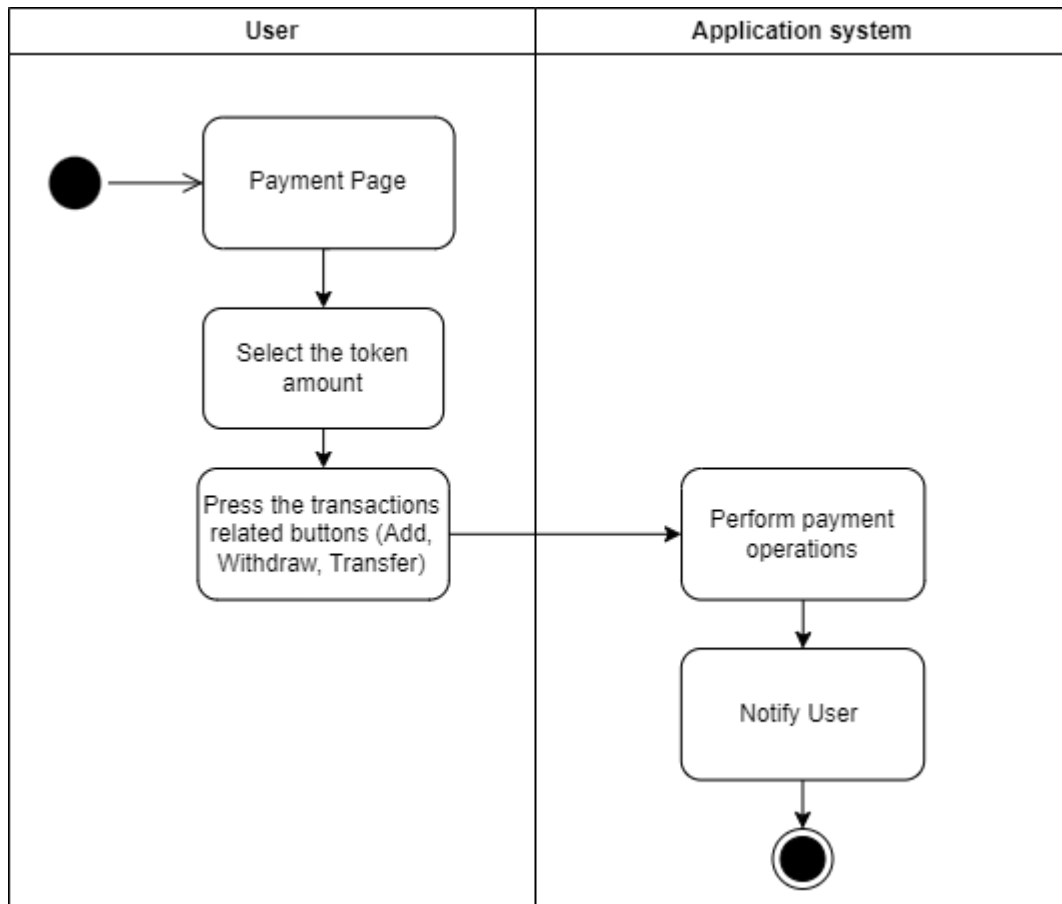


Figure 3.2.8: Activity diagram for use case “Payment Module”

Use Case: Purchase voucher

When a user opens up the purchase voucher page, firstly the user can select the voucher that they want to purchase and then proceeds to make payment. The system application system will then deduct funds from the wallet and also upload the order information to the database.



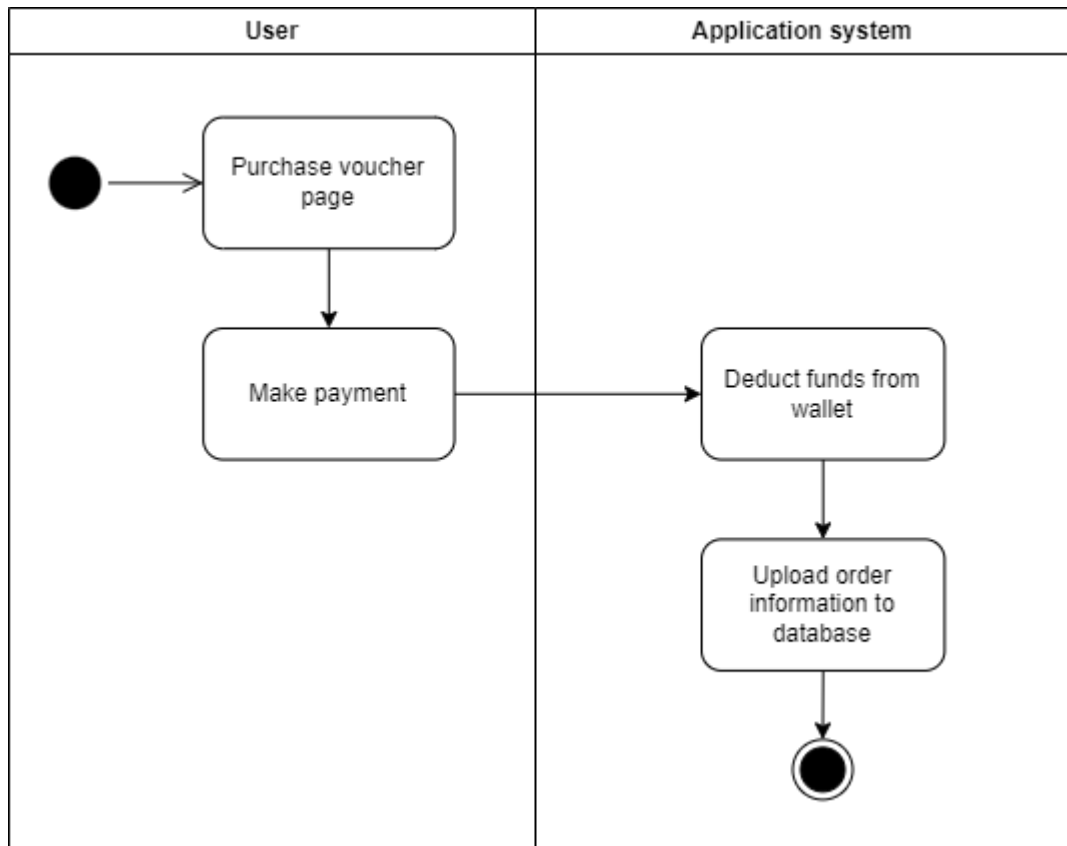


Figure 3.2.9 Activity diagram for use case “Purchase Voucher”

#### Use Case: FAQ page

When a user opens up the FAQ page, the system application system will then display all of the FAQ questions and answers to the user. This will help the user to get familiar with the operations of the application

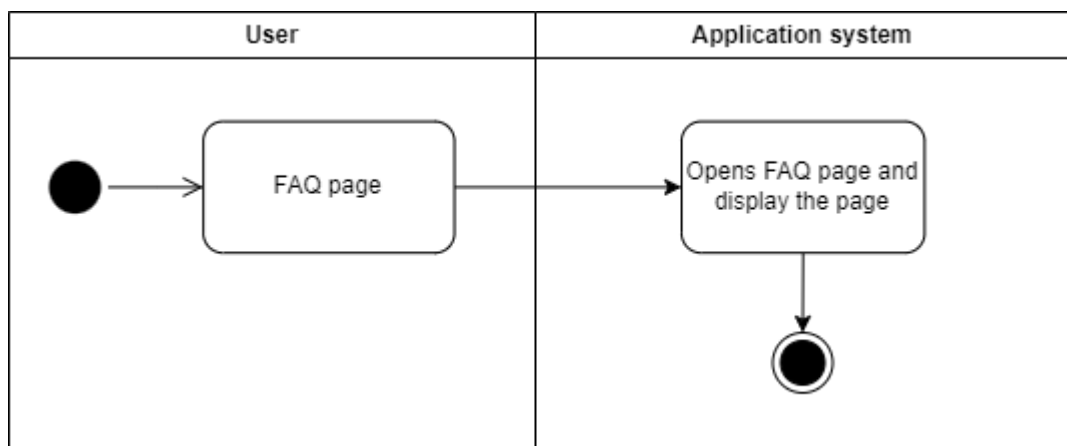


Figure 3.2.10: Activity diagram for use case “FAQ page”

Use Case: Provide Feedback

When a user wants to view data usage, the user will press on the provide feedback button. The user will input the feedback. After that, the feedback will be uploaded to the system. (Figure 3.11)

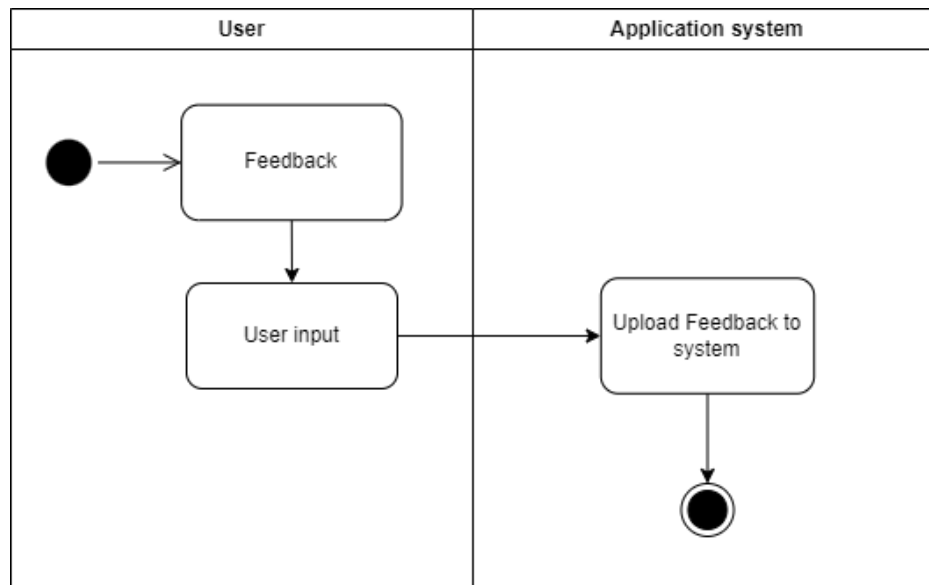


Figure 3.2.11: Activity diagram for use case “Provide Feedback”

Use Case: Sign Out

When a user press on the sign-out button, the system application system will then proceed to sign out the user and returns to the login page.

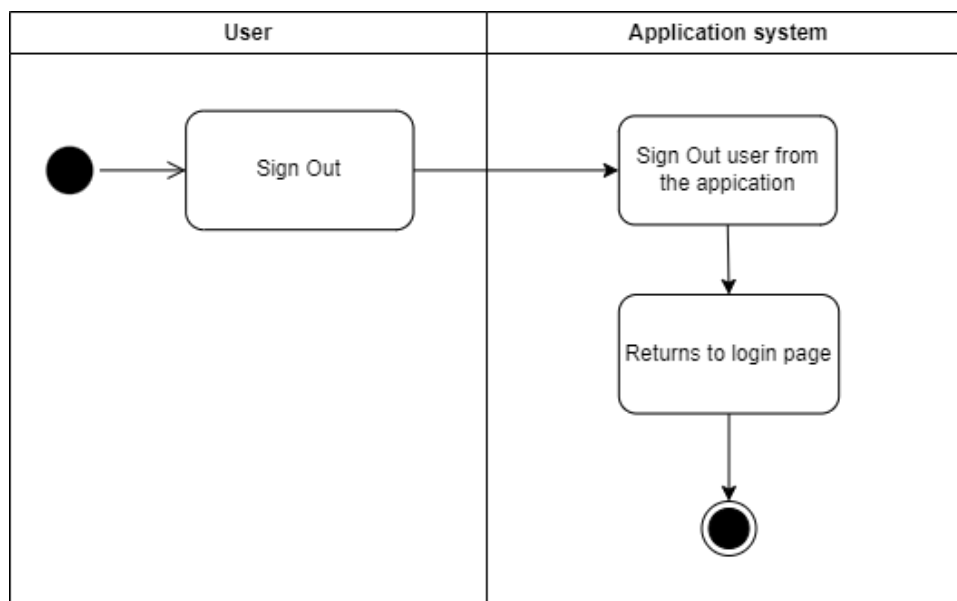


Figure 3.2.3: Activity diagram for use case “Sign Out

## Chapter 4 System Design

### 4.1 System Block Diagram

The system of this project consists of 4 modules, which are the connection module, nearby hotspot module, payment module, and also the settings module (Figure 4.1). Initially, a login module is required to authenticate the users. After logging in to the application, there will be 4 modules that the user can use. First, the connection module is the module that allows user to advertise their mobile hotspot to others. Alternatively, the connection module also allows the user to discover nearby available hotspot and then connect to the desired hotspot. Secondly, the application has a nearby hotspot module and this module works by displaying the nearby hotspot locations markers on the map. As a result, users can use the markers to locate and navigate to the location where a hotspot is available. Next, the application also consists of a payment module. The payment module in this system implements a smart contract to facilitate the transactions after the hotspot-sharing sessions. In addition, there is also a voucher catalogue provided where users are allowed to purchase vouchers by using the wallet token. Lastly, the settings module contains the sign-out functionality, the feedback form as well as a FAQ page for the user.

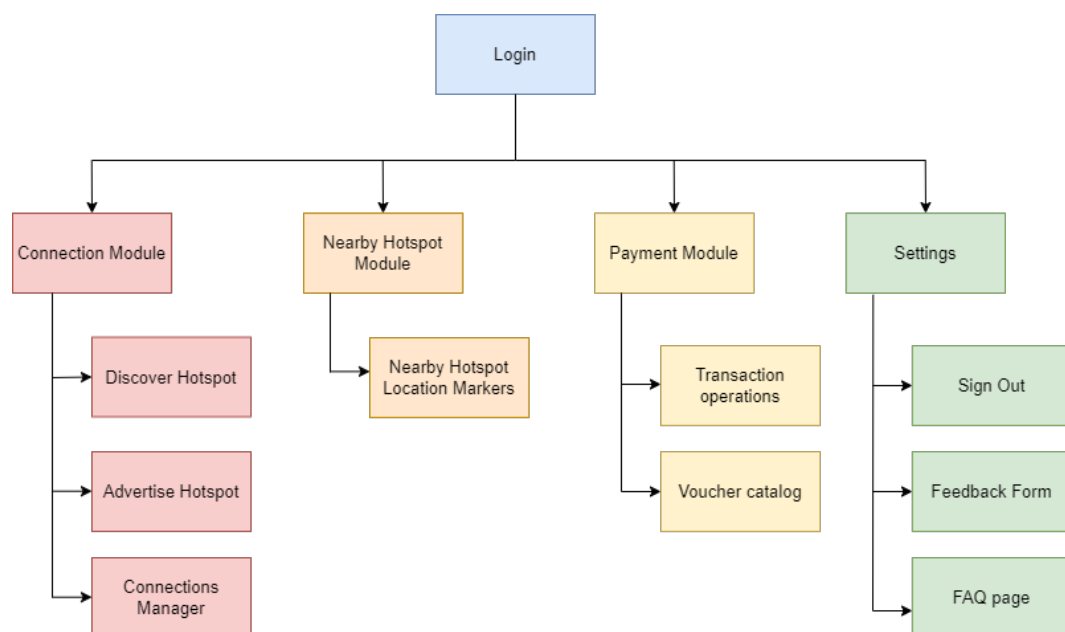


Figure 4.1: System Block Diagram

## 4.2 System Components Specifications

This section will cover and discuss the 3 most important modules in the system, which are the connection module, nearby hotspot module and payment module. The methodologies used and the general system flow in those modules will be discussed.

### 4.2.1 Connection Module

The Connection Module uses the Nearby Connections as the main medium to transfer the hotspot credentials between a hotspot advertiser and a hotspot discoverer. Nearby Connections is an API framework that enables seamless peer-to-peer communication between android devices, even without an internet connection. It establishes connections and exchanges data using wireless technologies such as Bluetooth and Wi-Fi Direct. The Nearby Connections is a great technology to be used in this connection module due to the peer-to-peer connection fashion. This allows the discoverer device, which does not have internet access to discover and obtain the hotspot credentials from the advertiser device. Figure 4.2.1 shows the general process that happened between an advertiser and a discoverer while connecting to each other.

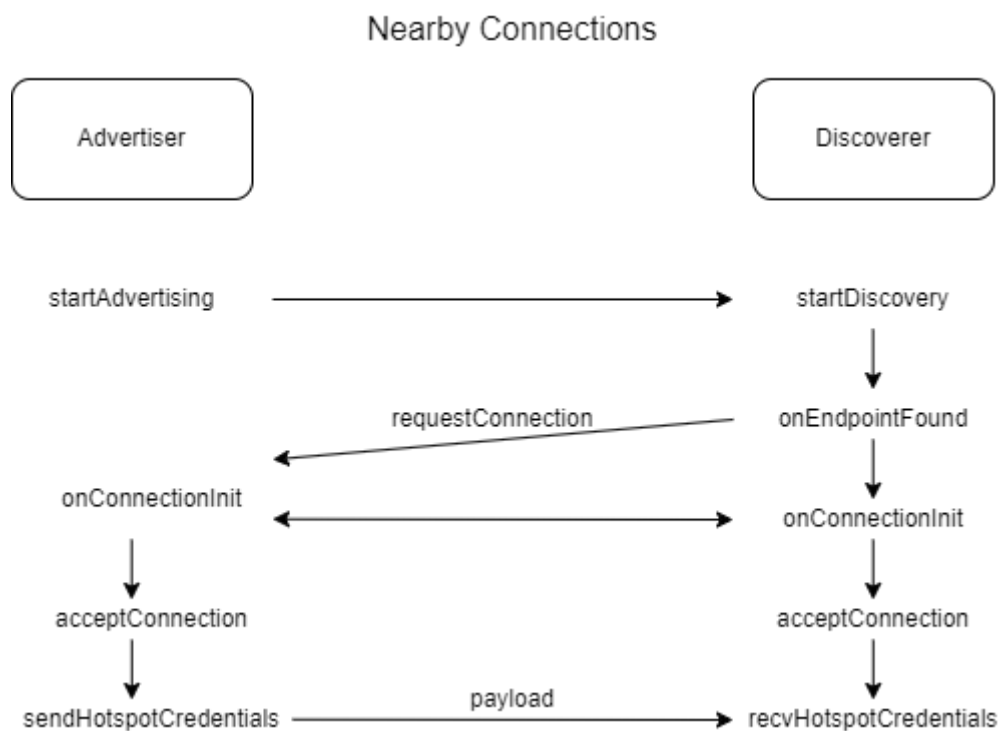


Figure 4.2.1: Process of the Nearby Connections

The flow of the process of the advertise hotspot function will be: First the hotspot advertiser will start advertising the availability of the hotspot by using the Nearby Connections advertising mode. At the same time, the location coordinates will be uploaded to the database. After that, the advertiser will then wait for an incoming connection from a discoverer. Once a connection is found, the advertiser can then accept or reject the connection. After successfully connecting with the discoverer, then the advertiser will wait for the discoverer to specify the hotspot duration they wanted to use. After receiving the desired hotspot duration, then the hotspot advertiser will then send over the hotspot credentials to the hotspot discoverer.

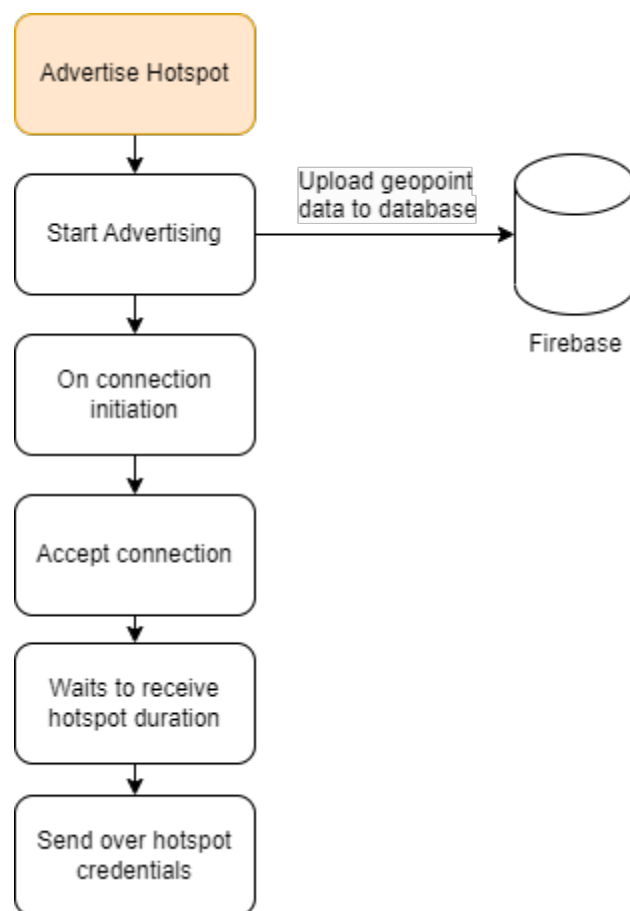


Figure 4.2.2: Block diagram of advertise hotspot function

The flow of process of the discover hotspot function will be: First the hotspot discover will start discovering nearby the available advertiser by using the Nearby Connections

discovering mode. After that, the discoverer will find and display the nearby advertiser in a list. The discoverer will then be able to request a connection to the advertiser. On the initiation of the connection, the discoverer can then accept or reject the connection. After successfully connecting with the advertiser, then the discoverer select send over the hotspot duration that they wanted to use. After sending the desired hotspot duration, then the hotspot discoverer will then wait for the advertiser to send over the hotspot credentials. Finally, the hotspot discoverer can successfully connect to the hotspot with the hotspot credentials given.

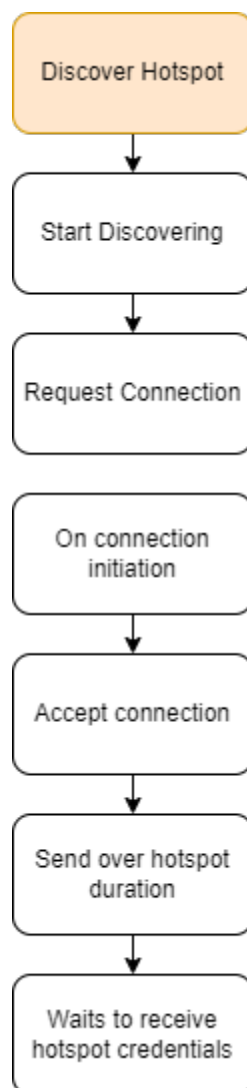


Figure 4.2.3: Block diagram of discover hotspot function

### 4.2.2 Nearby Hotspot Finder Module

The Nearby Hotspot Module uses the Google Map API as well as the Haversine formula to calculate and pinpoint the nearby available hotspot. The Haversine formula is used to calculate the distance between two points on the Google Maps. The Haversine formula is given below:

$$d = 2r \arcsin \left( \sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)} \right)$$

Let:

- $\varphi_1$  and  $\varphi_2$  be the latitudes of Point 1 and Point 2, respectively.
- $\lambda_1$  and  $\lambda_2$  be the longitudes of Point 1 and Point 2, respectively.
- $r$  be the radius of the sphere (e.g., Earth's radius).

Where:

- $d$  represents the distance between Point 1 and Point 2.

First, the application will retrieve the hotspot advertiser location data from the database. After that, the system will perform a filter on the hotspot advertiser location data. The Haversine formula is used to calculate and filters out the location data that is not in the 10km range from the device location. Besides that, the system will also filter out the location data that has a timestamp of 1 hour and above, hence only the hotspot advertiser location data that is advertising within 1 hour is being kept. After that, the system will then display the filtered location information by using the markers displayed on the map.

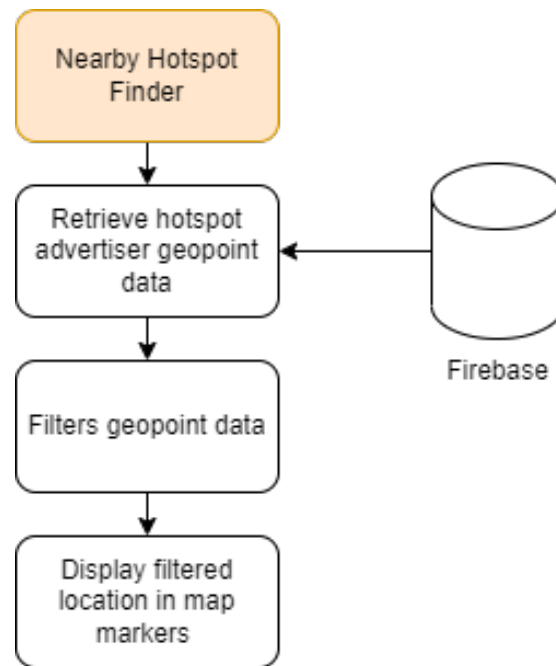


Figure 4.2.4: Block diagram of nearby hotspot finder function

#### 4.2.3 Payment Module

The payment module implements a smart contract to manage and automate the transactions in the application. The structure of the smart contract is a simple wallet contract that has a mapping for wallet address to token balance. Hence, every user in the app will be having a wallet address that will store the amount of token balanced associated with the address. Besides that, the smart contract also contains functions such as transfer, getBalance, addFunds, and also removeFunds. The transfer function is used to transfer the tokens between users' wallets (Figure 4.2.5). The getBalance function is to retrieve the token balance associated with the address and also the addFunds and removeFunds functions are to add or remove tokens to the wallet address respectively. The smart contract is being deployed in the Ethereum test network, named Sepolia test network. Hence, the application can access to the smart contract within the app using the Infura API. The Infura API connects the Flutter app to the Sepolia test network so that the application can access to the smart contract deployed in the application.



```
function transfer(address payable sender, address payable recipient, uint256 amount) external {  
    require(amount > 0, "Amount must be greater than zero");  
    require(balances[sender] >= amount, "Insufficient balance");  
  
    balances[sender] -= amount;  
    balances[recipient] += amount;  
  
    emit Transfer(sender, recipient, amount);  
}
```

Figure 4.2.5: transfer function in the smart contract

First, the application will initiate a Web3Client by using the Infura API to allow the application to connect and access the blockchain network. After that, when any transaction action is being performed in the application, the application will invoke a function in the smart contract by accessing the smart contract address through the Web3Client that are initiated earlier on.

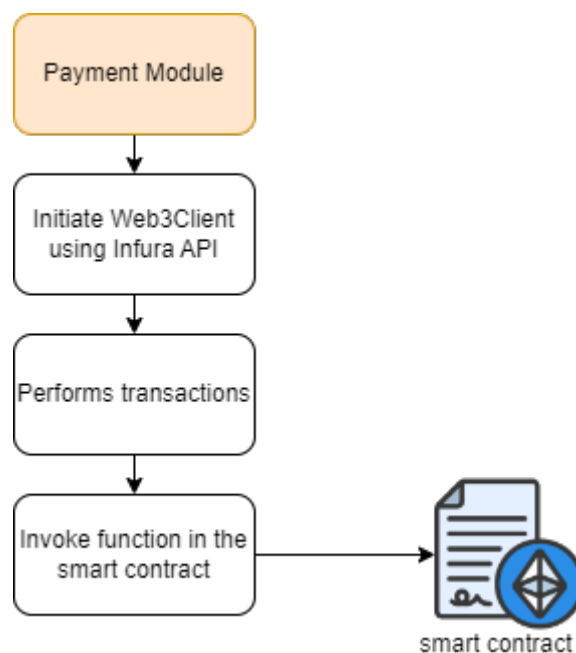


Figure 4.2.6: Block diagram of payment module

## Chapter 5 System Implementation

### 5.1 Hardware Setup

The hardware involved in this project is a computer and two Android smartphones. The computer is used to code and program the project's mobile application. The two smartphones are used for testing and deploying the mobile application developed. The table below shows the specifications of the computer (Table 3.1) and the smartphone (Table 3.2) and (Table 3.3)

Description	Specifications
Processor	Intel(R) Core(TM) i7-9700
Operating System	Windows 11
Graphic	NVIDIA GeForce GTX 1660 SUPER
Memory	16GB DDR4 RAM
Storage	512GB NVME SSD + 2TB SATA HDD

Table 5.1.1 Specifications of computer

Description	Specifications
Model	Honor View 20
Processor	Huawei Kirin 980
Operating System	Android 10, Magic UI 4.0
Graphic	Mali-G76 MP10
Memory	8GB RAM
Storage	256GB ROM

Table 5.1.2: Specifications of smartphone 1

Description	Specifications
Model	Honor 8X
Processor	Kirin 710 (12 nm)
Operating System	Android 10, EMUI 9.0
Graphic	Mali-G51 MP4
Memory	4GB RAM
Storage	128GB ROM

Table 5.1.3: Specifications of smartphone 2

## 5.2 Software Setup

The table below shows the list of software used (Table 3.2.3) in this project:

Software	Description
<b>Flutter</b>	The cross-platform SDK and framework used to develop the mobile application frontend
<b>Solidity</b>	Helps to implement smart contracts for in-app transactions
<b>Android Studio</b>	Built-in Android emulator that helps to deploy and test developed mobile application
<b>Visual Studio Code</b>	Main source-code editor for the project mobile application
<b>Firebase</b>	A Cloud-hosted database that is used as the backend database for the mobile application
<b>Infura</b>	Provides Infura Web3 API to link Flutter with the blockchain network
<b>Maps SDK for Android and Google Maps API</b>	Maps SDK provides map service to the mobile application, while Google Maps API provides API calls for displaying markers on the map

Table 5.2.1: List of software used

### 5.3 User Requirements

#### 5.3.1 Hardware Requirements

This project deliverable currently only supports Android devices with Android 6.0 and above OS.

#### 5.3.2 App Permissions

The following list shows the app permissions that the user must approve:

- Location
- Storage
- Full network access

#### 5.3.3 Connectivity Requirements

The following table shows the connectivity requirements for a user:

Connectivity	Requirement (Yes/No)
Wi-Fi/Cellular connection	✓
GPS service	✓
Bluetooth	✓

Table 5.3.1: User connectivity requirement

### 5.4 Setting and Configuration

This project requires preliminary setting and configuration to runs the application.

#### Setting up developer mode on android device

1. Open up device settings and turn on the developer option.
2. In the developer option, under debugging, turn on USB debugging and also install via USB (as shown in Figure 5.1).
3. After that, the user will be able to build and run the application in the mobile device via a USB connection for implementation and testing purpose.

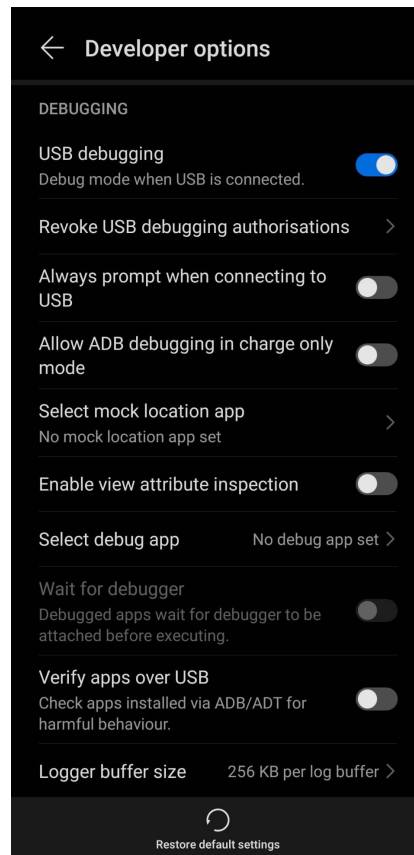


Figure 5.4.1: Developer options of mobile device

### Declaring required permissions in AndroidManifest.xml

The permission used in the mobile application must be declared in the AndroidManifest.xml in order for the mobile application to work as intended. Below shows the permissions declared in the AndroidManifest.xml.

```
<!-- Required for Nearby Connections and wifi api -->
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.READ_PRIVILEGED_PHONE_STATE"
    tools:ignore="ProtectedPermissions" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.PACKAGE_USAGE_STATS"
    tools:ignore="ProtectedPermissions" />

<!-- For Android 12+ support-->
<uses-permission android:name="android.permission.BLUETOOTH_ADVERTISE" />
<uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
```

Figure 5.4.2: Declaring permissions in the AndroidManifest.xml

### Installing related flutter dependencies

There are several Flutter packages that are being used in this project. The packages should be installed using 'pub get' in Visual Studio Code

```
dependencies:  
  android_flutter_wifi: ^0.1.0  
  cloud_firestore: ^4.5.0  
  connectivity_plus: ^3.0.5  
  cupertino_icons: ^1.0.2  
  curved_navigation_bar: ^1.0.3  
  firebase_auth: ^4.4.2  
  firebase_core: ^2.10.0  
  flutter:  
    sdk: flutter  
  flutter_dotenv: ^5.0.2  
  fluttertoast: ^8.2.1  
  geolocator: ^9.0.2  
  google_fonts: ^4.0.3  
  google_maps_flutter: ^2.2.5  
  google_sign_in: ^6.1.0  
  http: ^0.13.5  
  image_gallery_saver: ^1.7.1  
  image_picker: ^0.8.7+1  
  json_serializable: ^6.6.1  
  lottie: ^2.3.2  
  nearby_connections: ^3.3.0  
  network_info_plus: ^3.0.5  
  path_provider: ^2.0.14  
  plugin_wifi_connect: ^1.0.2  
  provider: ^6.0.5  
  qr_code_scanner: ^1.0.1  
  qr_flutter: ^4.0.0  
  request_permission: ^2.1.4  
  screenshot: ^1.3.0  
  sign_in_button: ^3.2.0  
  smooth_page_indicator: ^1.1.0  
  syncfusion_flutter_sliders: ^21.1.41  
  url_launcher: ^6.1.10  
  web3dart: ^2.6.1  
  wifi_info_plugin_plus: ^2.0.2  
  wifi_iot: ^0.3.18
```

Figure 5.4.3: List of the flutter dependencies used

### Setting up Firebase Cloud Firestore and Authentication

The following are the preliminary setup for Firebase Database and Authentication

1. Open the pubspec.yaml file in your project directory and add the 'firebase\_core', 'firebase\_auth', and 'cloud\_firestore' dependencies under the dependencies section.
2. Save the pubspec.yaml file and run flutter pub get in the terminal to fetch the dependencies.
3. Set up a Firebase project:
  - a. Go to the Firebase Console.
  - b. Click on "Add project" or select an existing project.
  - c. Follow the instructions to create a project in the Firebase Console.
4. Set up Firebase Authentication:
  - a. In the Firebase Console, go to the "Authentication" section.
  - b. Click on the "Set up sign-in method" button.
  - c. Enable the sign-in methods (e.g., email/password, Google, etc.).
5. Set up Firebase Cloud Firestore:
  - a. In the Firebase Console, go to the "Database" section.
  - b. Choose "Cloud Firestore"
  - c. Set up the database
6. Connect the Flutter app to Firebase by registering the app and download the 'google-services.json' file. After that, place the file in the 'android/app/src' directory
7. Finally, initialize the Firebase in the app by adding the relevant codes to start using the firebase services:

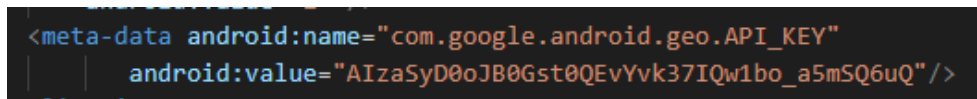
```
Future main() async {  
  WidgetsFlutterBinding.ensureInitialized();  
  await Firebase.initializeApp();  
  
  runApp(const MyApp());  
}
```

Figure 5.4.4: Initialization of Firebase

### Setting up Google Maps API

The following are the preliminary setup for Google Maps API:

1. Open the 'pubspec.yaml' file in the project directory and add the 'google\_maps\_flutter' dependency under the dependencies section
2. Save the pubspec.yaml file and run flutter pub get in the terminal to fetch the dependency.
3. Obtain an API key from the Google Cloud Platform Console
  - Go to the Google Cloud Platform Console.
  - Create a new project or select an existing project.
  - Enable the "Maps SDK for Android" and "Maps SDK for iOS" APIs for your project.
  - Create an API key for your project.
4. After that, open the AndroidManifest.xml file and add the API key in the file with the <meta-data> tag.



```
<meta-data android:name="com.google.android.geo.API_KEY"
           android:value="AIzaSyD8oJB8Gst8QEYvYvk37IQw1bo_a5mSQ6uQ"/>
```

Figure 5.4.5: Setting up Google Maps API with API key

5. Finally, save the configuration file and the Google Maps API is all set up.

### Setting up Infura API

The following are the preliminary setup for Infura API:

1. First, create an Infura Account at (<https://infura.io>) and log in. Then, create a project and obtain an API key.
2. Open the 'pubspec.yaml' file in the project directory and add the 'web3dart' and 'http' under the dependencies section.
3. Then, run 'flutter pub get' to fetch the dependencies.
4. Then, implements the Infura API connection by importing the 'web3dart' and 'http' packages and create an instance of 'Web3clinet' by providing the Infura API URL and the API key (Diagram 5.6).



```
void initial() {
    httpClient = http.Client();
    String ethEndpoint =
        "https://sepolia.infura.io/v3/6e68f54f25b54035b4167dc60f24695c";
    web3client = Web3Client(ethEndpoint, httpClient);
}
```

Figure 5.4.6: Setting up the Infura API using Web3Client

5. After that, the Infura API is all set up and is ready to make various API calls to the blockchain network.

### Setting up the smart contract

The following are the preliminary setup for the smart contract:

1. First, go to remix IDE and create a .sol file that will contain the codes for the smart contract. After that, write the smart contract and provide the necessary function in the contract.
2. Next, after finish coding the smart contract, compile the smart contract and then the smart contract is ready to be deployed.
3. To deploy the smart contract, first select the Injected Provider – MetaMask as the environment to be deployed (MetaMask required) and select press deploy.
4. After that, the smart contract will be deployed into the Sepolia test network, and a contract address will be provided.
5. Then, place the ‘abi.json’ file under the ‘assets’ folder and invoke the file to get the contract details in order to interact with the smart contract (Figure 5.4.6).
6. The smart contract is then all set up and is ready to interact with the flutter app.

```
Future<DeployedContract> getDeployedContract() async {
    String abi = await rootBundle.loadString("assets/abi.json");
    final contract = DeployedContract(ContractAbi.fromJson(abi, "Wallet"),
        EthereumAddress.fromHex(contractAddr!)); // DeployedContract
    return contract;
}
```

Figure 5.4.6: Calling the smart contract in flutter app

```
contract Wallet {

    string public NYCToken;
    string public NYC;

    mapping(address => uint256) private balances;

    event Transfer(address indexed from, address indexed to, uint256 amount);
    event Balance(address indexed account, uint256 balance);

    function transfer(address payable sender, address payable recipient, uint256 amount) external {
        require(amount > 0, "Amount must be greater than zero");
        require(balances[sender] >= amount, "Insufficient balance");

        balances[sender] -= amount;
        balances[recipient] += amount;

        emit Transfer(sender, recipient, amount);
    }

    function getBalance(address account) external view returns (uint256) {
        return balances[account];
    }

    function addFunds(address account, uint256 amount) external {
        require(amount > 0, "Amount must be greater than zero");

        balances[account] += amount;

        emit Transfer(address(0), account, amount);
    }

    function removeFunds(address account, uint256 amount) external {
        require(amount > 0, "Amount must be greater than zero");
        require(balances[account] >= amount, "Insufficient balance");

        balances[account] -= amount;

        emit Transfer(account, address(0), amount);
    }
}
```

Figure 5.4.7: Contents of the smart contract

### 5.5 System Operation

In this project, the application is named as “Share&Link”. This application emphasizes on the sharing of hotspot connection to the other in need and linking the hotspot connection together to provide a greater network coverage to the area. The logo of the application is shown in Figure 5.5.1 below.



Figure 5.5.1: Logo of “Share&Link” application.

#### 5.5.1 Login Functionality

Upon launching the application, users are will be shown with a splash and also an onboarding screen as shown in Figure 5.5.1 and Figure 5.5.2 below, which will provide users an insight about what the application is about and what are the features provided in the application.



Figure 5.5.1: Application Splash

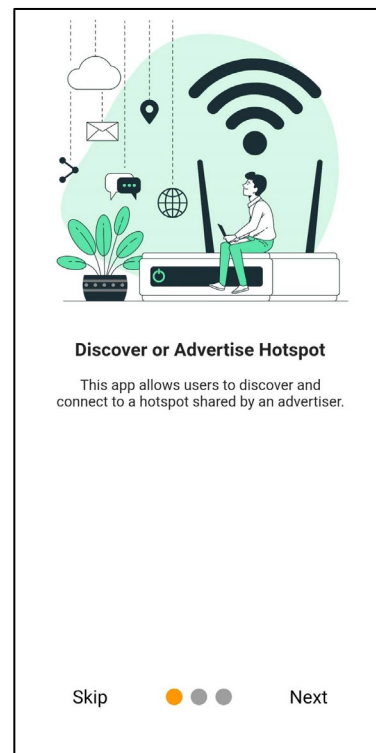


Figure 5.5.2: Onboarding screens

After that, the user will be redirected to the login page as shown in Figure 5.5.3 below. On the login page, users are required to input their login credentials, which can be the email and password combination, or also through sign in through Google Account. The app uses Firebase Authentication to perform the authentication of the login credentials, ensuring the credentials input are both correct and correspond to an existing account.

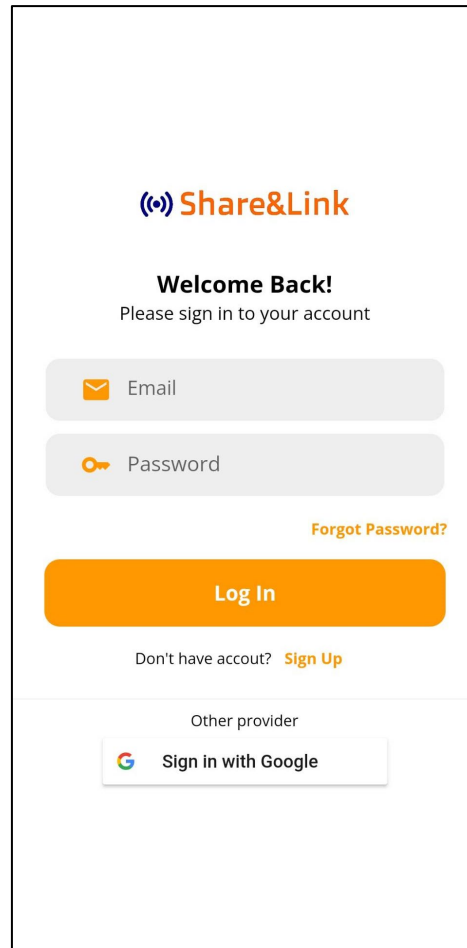


Figure 5.5.3: Application's login page

When a user inputs incorrect login credentials, the app will display an error message notifying the user that they have input the wrong login credentials. Moreover, if the user forgot their password, a user could press the forgot password button on the login page to navigate to the login password page as shown as Figure 5.5.4. On that page, users are required to input their email address and press the send reset password button.

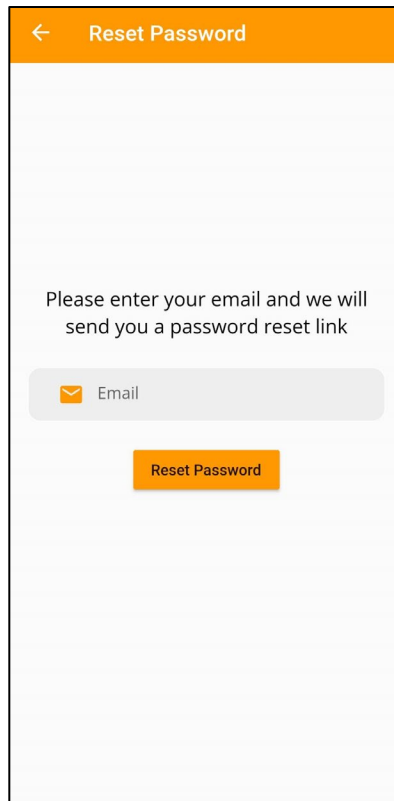


Figure 5.5.4: Application's reset password page

After that, an email which contains the password reset link will be sent to the email address and the user can reset their password (Figure 5.5.5).

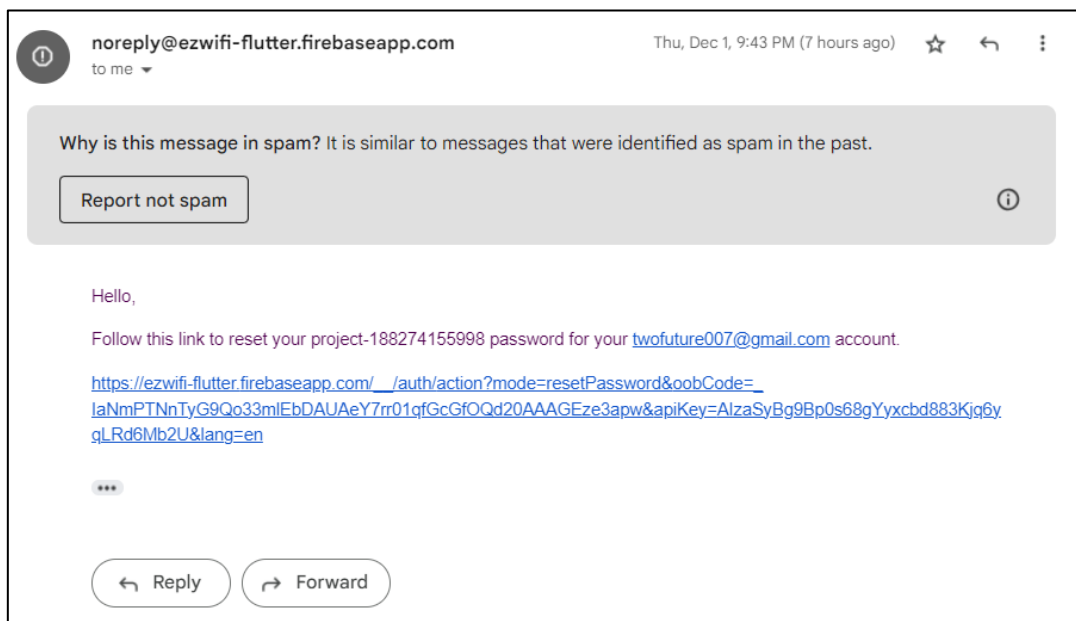


Figure 5.5.5: Email with a reset password link

## CHAPTER 5 SYSTEM IMPLEMENTATION

Alternatively, if a user does not have an account, the user can press on the Sign Up button to register for a new account. Users are required to sign up by key in the email address and password in the process. The system will then check if there are existing accounts associated with the email address. If there are no matching records found, then the user account will be successfully created. (Figure 5.5.6)

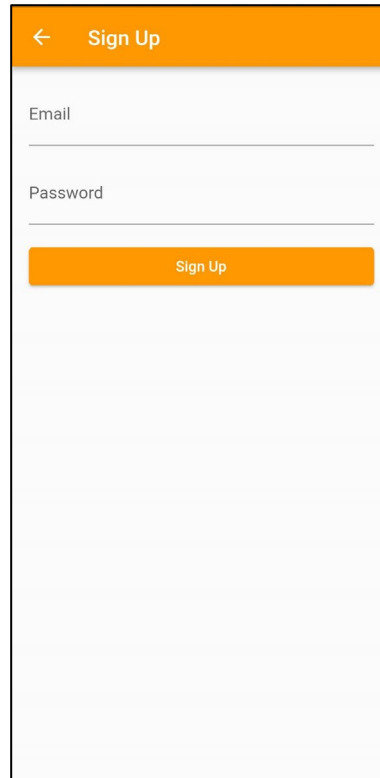
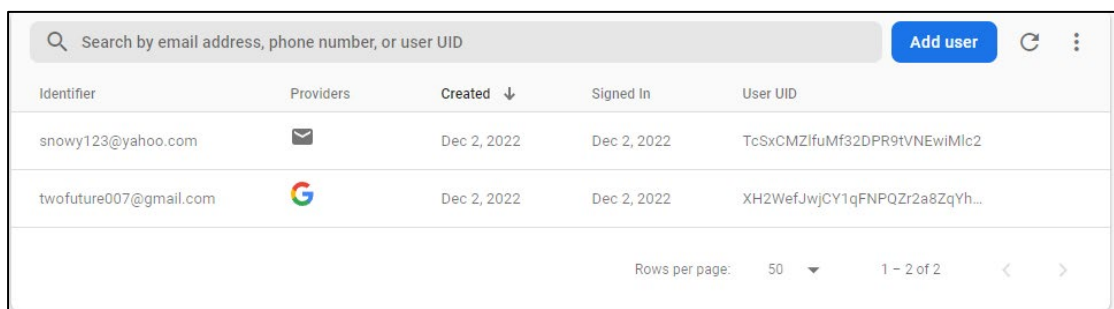


Figure 5.5.6: Application's Sign Up page

All of the authentication data will then be stored in the Firebase Authentication as shown in Figure 5.5.7.





Identifier	Providers	Created ↓	Signed In	User UID
snowy123@yahoo.com		Dec 2, 2022	Dec 2, 2022	TcSxCMZIfuMf32DPR9tVNEwiMlc2
twofuture007@gmail.com		Dec 2, 2022	Dec 2, 2022	XH2WefJwjCY1qFNPQZr2a8ZqYh...

Figure 5.5.7: Example of stored user login credentials

### 5.5.2 Advertising and Discovering of Hotspot

After logging in to the application, the main page which is the connection page (Figure 5.5.8). On this page, two main actions can be done by the user, which are advertise the hotspot, or to discover the hotspot. The advertise hotspot function is designated to let the user who has access mobile data to share their hotspot to the other who needs internet access, while the discover hotspot function is designated to let user who are looking for a hotspot to find a hotspot and gain internet access.

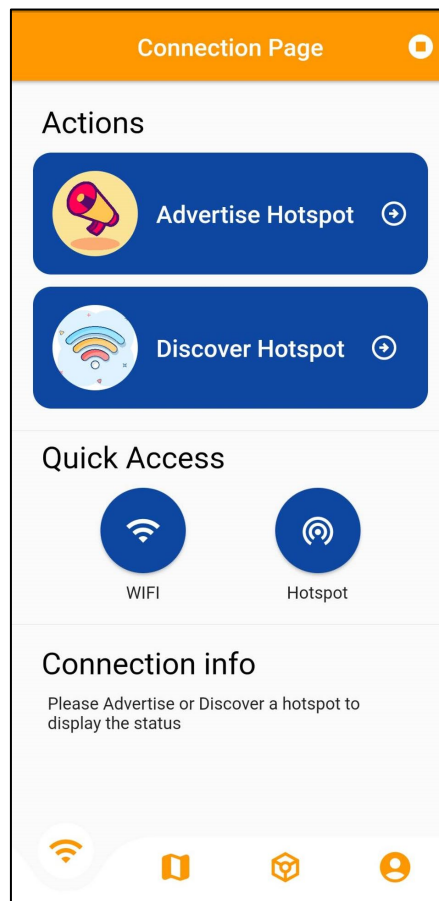


Figure 5.5.8: Application connection page

If the user press on the advertise hotspot button, the user will start advertising the availability of the hotspot (Figure 5.5.9). The application will advertise the availability by using the nearby connection API. At the same time, the advertiser's location will be uploaded to the database. On the other hand, if the user press on the discover hotspot, the user will start discovering available hotspot (Figure 5.5.10).

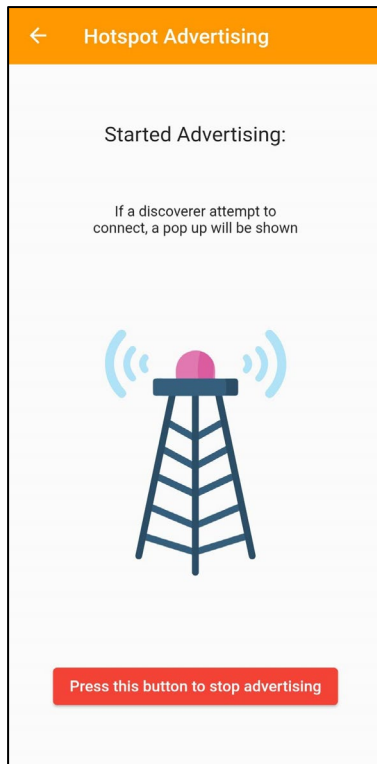


Figure 5.5.9: Advertising hotspot

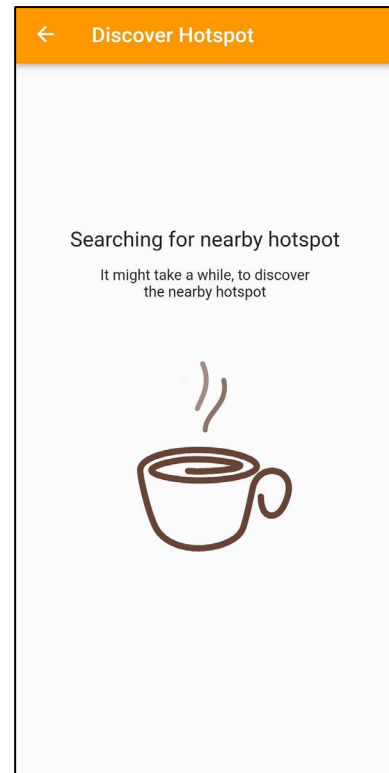


Figure 5.5.10: Discovering hotspot

At the discovering hotspot side, upon the app successfully found a nearby advertiser that is advertising their availability of the hotspot, the list of advertisers will be displayed in a list (Figure 5.5.11). Then, the hotspot discoverer can choose and connect to the hotspot advertiser.

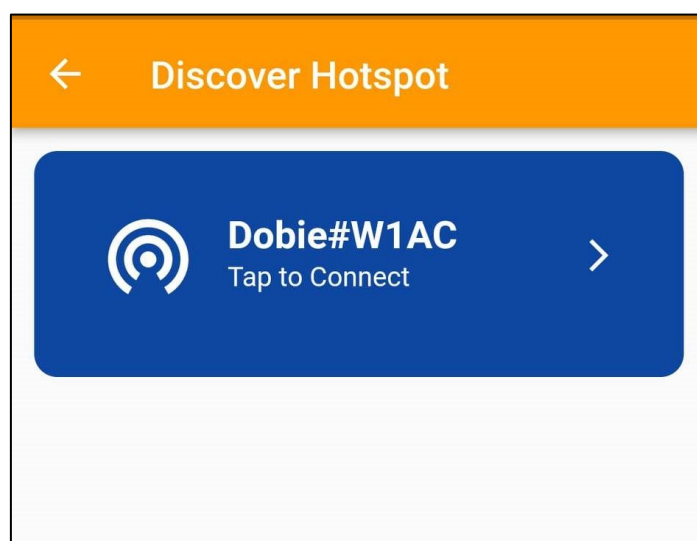


Figure 5.5.11: List of nearby available advertiser



## CHAPTER 5 SYSTEM IMPLEMENTATION

After the hotspot discover chooses and connects to the desired hotspot advertiser, the discoverer will attempt to initiate a connection to the advertiser. The advertiser will then receive the connection request from the discoverer and the user can choose to accept or reject the incoming request (Figure 5.5.12).

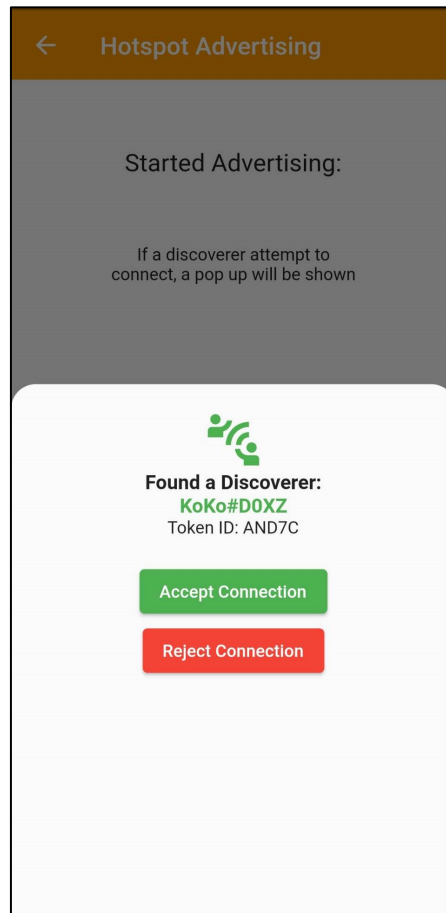


Figure 5.5.12: On connection initiation between discoverer and advertiser

If the user chooses to reject the connection, the connection will be dropped, and the user will then return to the main page. On the other hand, if the user chooses to accept the connection, both parties will then accept the connection and both parties are ready to transfer data to each other.

Again, the transferring part will be divided into two parties, which are the hotspot discoverer and also the advertiser. First, the hotspot discoverer will be prompted to send over the desired duration that they want to use the hotspot to the advertiser (Figure 5.5.13).

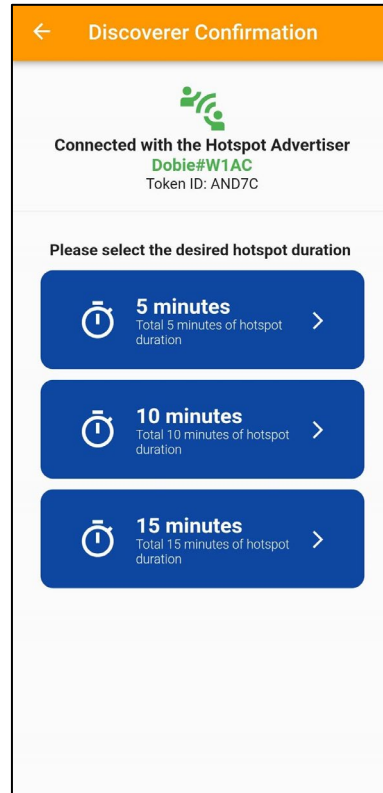


Figure 5.5.13: Process of sending the hotspot duration

At the hotspot advertiser side, the advertiser will first wait for the discoverer to send over the desired duration they wanted to use (Figure 5.5.14). If the discoverer does not send over the duration he wanted to use, then the process will not proceed. After receiving the duration specified by the discoverer, then the system will track the hotspot duration and then proceeds to let the user to send over their hotspot credentials to the discoverer (Figure 5.5.15).

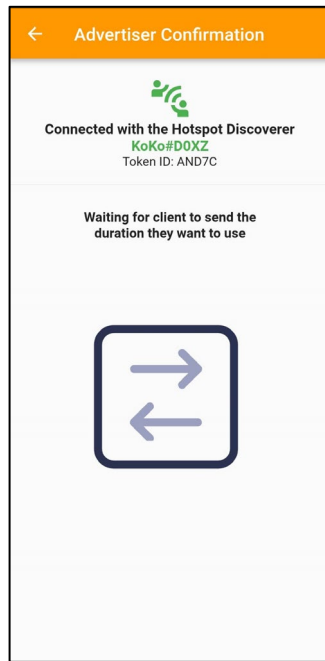


Figure 5.5.14: Incoming data page

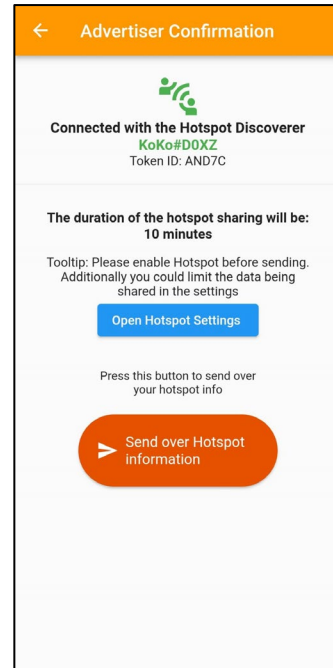


Figure 5.5.15: Send hotspot credentials page

After the advertiser press the send over hotspot information to the discoverer, the hotspot credentials will then be sent over to the discoverer. The information about the hotspot will then be displayed and a hotspot QR code be generated to help the user to connect to the hotspot more easily (Figure 5.5.16)



Figure 5.5.16: Hotspot information page

Finally, when the user returns to the main page, a timer will start countdown the duration that is specified in the process earlier on (Figure 5.5.17). Then user can then use the internet access within the duration specified. After the countdown timer is up, the hotspot will be turned off, and a notification will be sent to notify both the discoverer and also the advertiser.

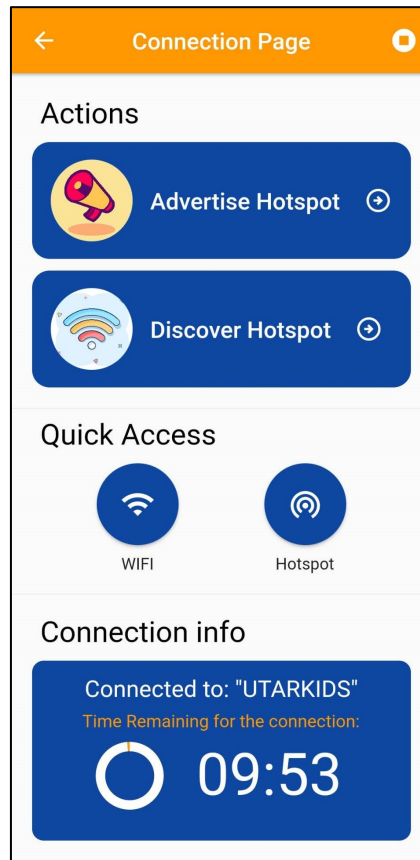


Figure 5.5.17: Countdown timer for the hotspot duration

After the hotspot advertiser has successfully shared the hotspot with the hotspot discoverer, the system will then execute the codes that will invoke the smart contract methods to transfer the funds from the discover wallet to the advertiser wallet. Hence, the hotspot advertiser will be rewarded with tokens upon successfully sharing the hotspot with others.

### 5.5.3 Nearby Hotspot Finder

When the user taps on the nearby hotspot finder, which is the map icon on the bottom navigation bar, the user will then be redirected to the nearby hotspot finder page. On the nearby hotspot finder page, a nearby hotspot finder map will be displayed. The nearby hotspot location information is fetched from the firebase database and filtered. Only the relevant hotspot location which is within 10km and also within 1 hour will be captured. After that, the filtered location information will be converted into markers and displayed on the map as shown as Figure 5.5.18.

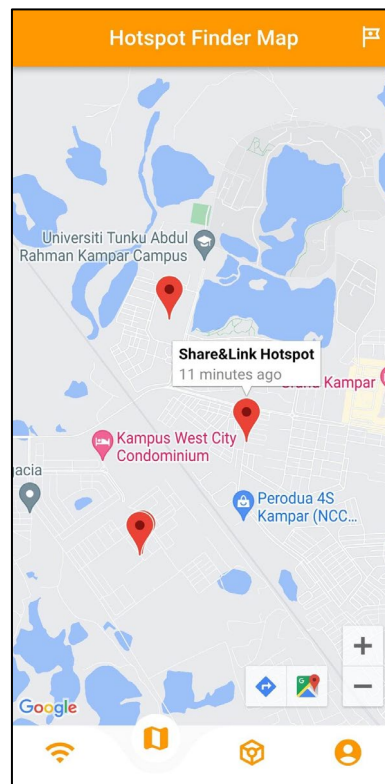


Figure 5.5.18: Nearby Hotspot Finder Map

Additionally, there is a button on the top right corner of the page that will display a tooltip where it will tell users how many available nearby hotspots that are around the user.

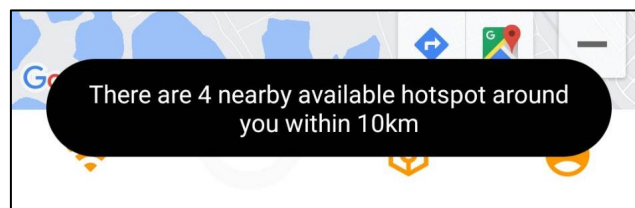


Figure 5.5.19: Tooltip for nearby available hotspot

## CHAPTER 5 SYSTEM IMPLEMENTATION

Users can then tap on the markers and navigate to the hotspot location by using the Google Maps application on their smartphones (Figure 5.5.20).

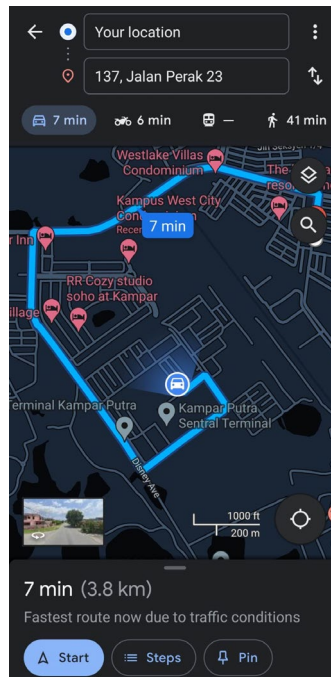


Figure 5.5.20: Navigate to the hotspot location using Google Maps

### 5.5.4 Payment Module

When the user taps on the payment module, which is the wallet icon on the bottom navigation bar, the user will be redirected to the wallet page. On the wallet page, the total balance of the wallet token will be displayed to the user. Additionally, there are a few transaction-related actions that can be done by the user. The actions included are payment to another account, add funds and remove funds. Besides that, there is a slider on the page that let users select the amount of token to be made in the transactions. (Figure 5.5.21). However, these actions are for the manual transaction only. After sharing the hotspot, the token transaction is executed automatically with the aid of the smart contract. A number of tokens from the discoverer wallet will be transferred to the advertiser wallet automatically.

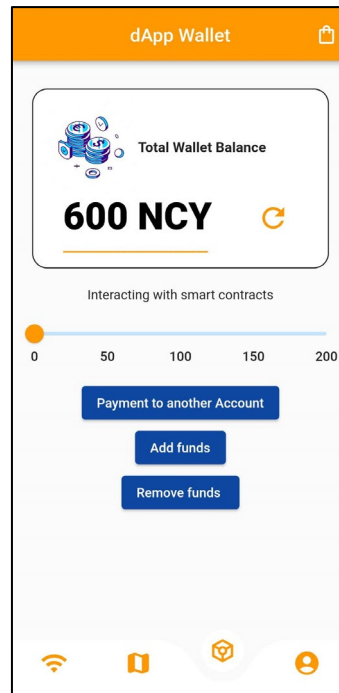


Figure 5.5.21: Application's wallet page

After the user performs the transactions of the token, a block which includes information such as transaction hash and methods called associated with the smart contract will be created (Figure 5.5.22). All of the blocks created represent each transaction that happened upon the execution of the smart contract, hence facilitating the transaction process in the application.

Sepolia Testnet

Search by Address / Txn Hash / Block / Token

Etherscan

Home Blockchain Tokens NFTs Misc

Contract 0xdC89fA31ac8BD15e557466A4c7246Fc12e747264

Overview

ETH BALANCE

0 ETH

More Info

CONTRACT CREATOR

0x38e3D6...f2D0a4BC at txn 0x6df6c833904e265d7...

Multi Chain

MULTICHAIN ADDRESSES

N/A

Transactions

Token Transfers (ERC-20) Contract Events

Latest 25 from a total of 45 transactions

Transaction Hash	Method	Block	Age	From	To	Value	Txn Fee
0x5b135d7e6fe744e8e...	0x50d6368c	3415772	1 day 3 hrs ago	0x38e3D6...f2D0a4BC	0xdC89fA...2e747264	0 ETH	0.00002948
0xeda47beb4485c662...	0x50d6368c	3403013	3 days 2 hrs ago	0x38e3D6...f2D0a4BC	0xdC89fA...2e747264	0 ETH	0.00004423
0xe252f66e13c1b3bd5...	0x50d6368c	3403008	3 days 3 hrs ago	0x38e3D6...f2D0a4BC	0xdC89fA...2e747264	0 ETH	0.00004452
0x34abfba3168c8f9d1...	Transfer	3403005	3 days 3 hrs ago	0x38e3D6...f2D0a4BC	0xdC89fA...2e747264	0 ETH	0.00005342
0xc457f405602a02f42...	0x50d6368c	3401939	3 days 6 hrs ago	0x38e3D6...f2D0a4BC	0xdC89fA...2e747264	0 ETH	0.00002968

Figure 5.5.22: Blocks creation on the smart contracts in the blockchain

Additionally, a user can use the token to make purchases on vouchers such as restaurant voucher, spa voucher, and movie voucher. Within the catalogue page, users can make an order by pressing the order button. (Figure 5.5.23)

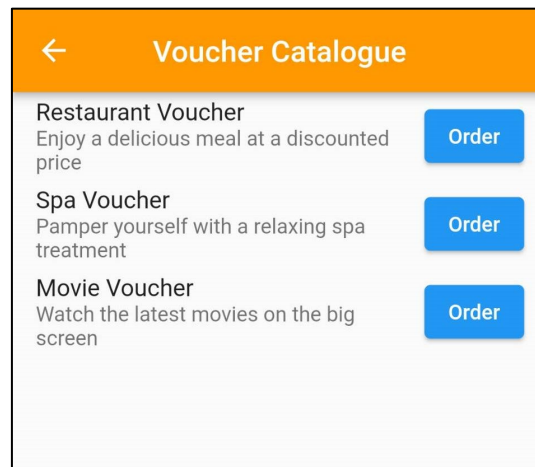


Figure 5.5.23: Voucher Catalogue Page

After the user pressed the order button, a number of tokens will be deducted from the user wallet. Then the order information will be uploaded to the database as shown in Figure 5.5.24.

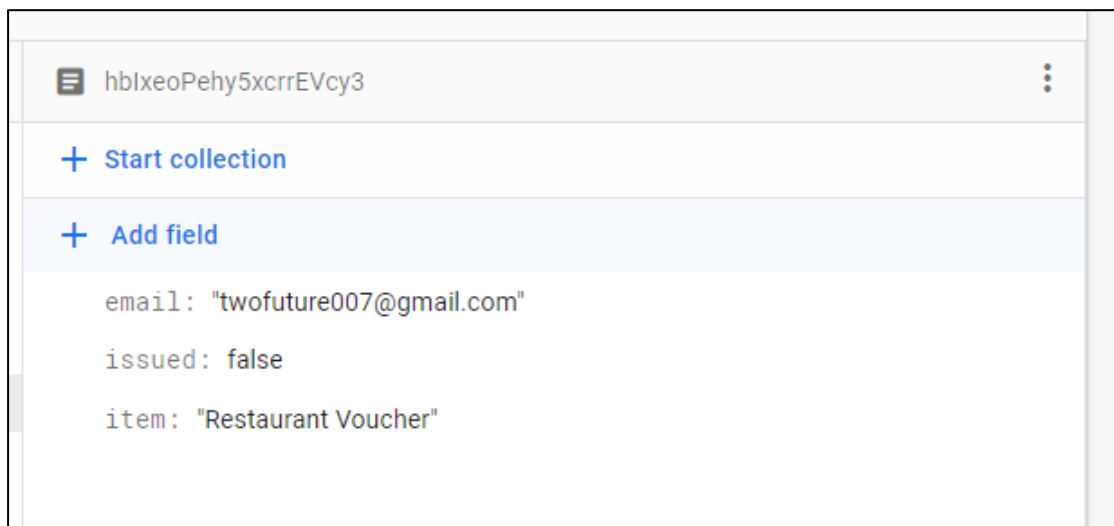


Figure 5.5.24: Example of the order information in database



### 5.5.5 Settings page

When the user taps on the settings page, which is the profile icon at the bottom navigation bar, the user will be redirected to the settings page (Figure 5.5.25). On the settings page, the information of the profile such as the picture and the email address will be displayed. There are three buttons on the settings page, which are the feedback button, the FAQ button and also the sign out button.

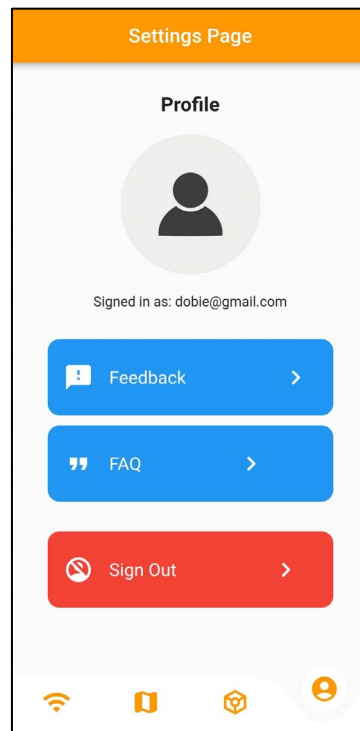
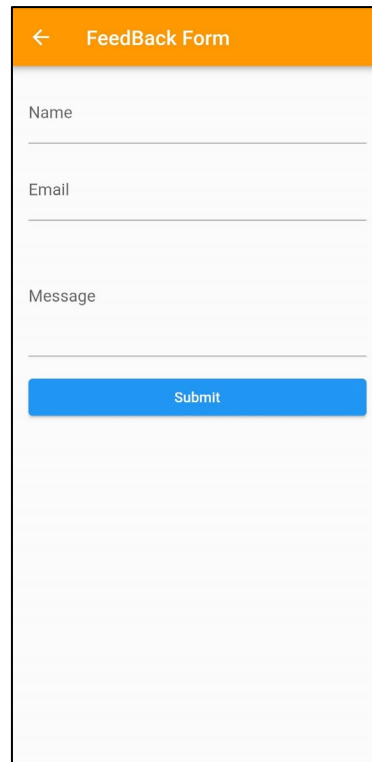


Figure 5.5.25: Application's settings page

When a user press on the feedback button, the user will be redirected to the feedback page. On the feedback page, the user is allowed to provide feedback by key in the name, email address, and also the feedback message (Figure 5.5.26). After the user press submit button, the feedback will be uploaded and stored in the database.



← FeedBack Form

Name

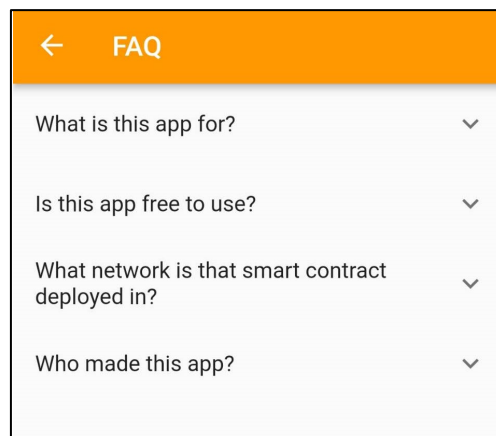
Email

Message

Submit

Figure 5.5.26: Application's Feedback page

When a user press on the FAQ button, the user will be redirected to the FAQ page. On the feedback page, a few frequently asked questions will be given to help the users understand more about the application and what does the application do (Figure 5.5.27).



← FAQ

What is this app for? ▼

Is this app free to use? ▼

What network is that smart contract deployed in? ▼

Who made this app? ▼

Figure 5.5.27: Application's FAQ page

Finally, when the user presses the sign out button, the user will be signed out and redirected back to the login page.

### 5.6 Implementation Issues and Challenges

The following implementations issues and challenges are faced during the development and implementation of the project:

- **Learning of new programming languages and tools.** The programming language used in this project is mostly Dart while developing the Flutter application and also Solidity for developing the smart contract in the blockchain network. Moreover, the tools used in this project include Firebase, Google Map API and also Infura API. Due to the lack of hands-on experience in the programming language and tools mentioned above, the time consumed to develop the project is increased significantly. The technique to integrate all of the components together is one of the biggest challenges in this project.
- **Lack of resources to learn.** Another challenge that is being encountered while developing the system is the lacking reliable resources to learn. Due to the nature of blockchain and smart contract is not widely adopted by the public, the resources available out there is very limited. Blockchain and smart contract technology are still in their early stages, and it is difficult to implement a complex system in this project. Besides that, due to the lack of resource to learn and implements the smart contract in this project, a huge amount of time is spent learning the technology from scratch. Greater effort is also put into this project by asking seniors or supervisors who have used similar technologies before.
- **Device compatibility and performance issues.** It can be difficult to guarantee that the application functions without hiccups on a variety of hardware configurations, screen sizes, and operating systems. In order to prevent problems like latency, sluggish rendering, and high battery consumption, performance optimization is important to make sure all of the devices will run the application smoothly.

- **Failure to implement the in-app network statistic and settings.** The inability to provide the in-app network statistics and settings is one of the biggest implementation problems that is being undergone during the development process of the application. Due to the depreciated Wi-Fi Manager Android API in Android 10, developers are no longer to access to the android API. Besides that, android also limits developers to gain access to the overall network statistics in the device, which includes the total upload and download data in the device. Hence, it was unable to develop a system that is more comprehensive that could provide users to configure the amount of hotspot data they want to consume, and also the inability to display the network statistics used when connected to the hotspot.
- **Data synchronization issue.** The nearby hotspot finder map and also the payment module requires a stable internet connection in order to use those functions. Due to the system using the database hosted on the Firebase, hence internet connectivity is required while using the nearby hotspot finder map in order to use the functionality. Besides that, internet connectivity is also required while using the payment module. Since the payment module is interacting with the smart contract for every transaction. Hence, a stable internet connection is required during the process. If a user loses the internet connection, the user might not be able to get the latest markers on the nearby hotspot as well as unable to make transactions during that time.

### 5.7 Concluding Remarks

In short, Chapter 5 has provided a comprehensive overview of the system implementation phase, including the development of the system, the systems operations with relevant screenshots, and the challenges encountered. This chapter focuses on the integration, design, and functionality of the numerous modules and components that is included in the system.

The difficulties and problems encountered during the implementation phase have been exhaustively analysed, providing valuable insight into the project's complexity and nuances. These experiences have shaped the approach and contributed to the system's refinement, ensuring that it meets the project's objectives and specifications.

In the next chapter, the testing and evaluation process on the implemented system will be carried out in Chapter 6. By carefully evaluating the system's performance, functionality, and usability, it is to be ensured that the system satisfies the expectations and has a significant impact in its intended context.

## **Chapter 6 System Evaluation and Discussion**

### **6.1 System Testing and Performance Metrics**

System testing and performance is the process of verifying and validating the outcome of the system developed. The objective of system testing is to make sure that the system complies with its specifications and requirements defined and operates smoothly. System testing is primarily concerned with ensuring that the system performs as anticipated and complies with user and business requirements. Defects and problems that may arise when several system components interact with one another are made easier to find. Following the process of testing, the results are examined, and any problems are communicated to the development team so they may be fixed. This will assure that the system is prepared for distribution to end users.

The application is created to provide hotspot sharing and discovering for the users, and additionally integrating the nearby hotspot finder map and also smart contract payment module to facilitate the transactions. However, during the development process, various problems such as unresponsive functions and errors might occur. As a result, it is essential to perform system testing that tests out the application under different scenarios to make sure the application is functioning in the best condition.

The type of system testing that will be conducted in this part is the unit testing. In unit testing, a checklist that contains a list of items and components in the application will be prepared. Additionally, the checklist often includes a list of test cases or actions that must be taken to verify a specific system function or component.

### **6.2 Testing Setup and Result**

A checklist will be used to perform the unit testing. This checklist will ensure that all the necessary items in the application are tested and no important aspects are overlooked during the testing process. The testing part will be breakdown into different sections, with each section representing one of the main modules in the application.

**Login Page Testing**

After launching the application, the system should display the login page to the user and clicking each of the buttons must redirect to the respective page or function.

No.	Test Case	Expected Result	Status
1.	Launch the application	Show splash screen and onboarding screens	Passed
2.	Login Page	Redirect to login page after onboarding screen	Passed
3.	Input email and password	Able to Input text into both text field.	Passed
		Display error for wrong input	Passed
4.	Press 'Login' Button	Redirect to main page	Passed
5.	Press 'Forgot Password' button	Redirect to reset password page	Passed
6.	Press 'Sign Up' button	Redirect to sign up page	Passed
7.	Press 'Sign in with Google' button	Sign in with google credentials and redirect to main page	Passed

Table 6.2.1: Test results for Login Page

**Forgot Password Page Testing**

After clicking the 'Forgot Password' button, the system will display the forgot password page to the user, and each of the buttons must redirect to the respective page or function.

No.	Test Case	Expected Result	Status
1.	Input email	Able to input text into text field.	Passed
2.	'Reset password' button	Able to send reset password link to respective email.	Passed
3.	Reset password	Able to reset password with the link provided	Passed

4.	Press 'Back' button	Redirect to login page	Passed
----	---------------------	------------------------	--------

Table 6.2.2: Test results for Forgot password page

**Sign Up Page Testing**

After clicking the 'Forgot Password' button, the system will display the Sign Up page to the user, and each of the buttons must redirect to the respective page or function.

No.	Test Case	Expected Result	Status
1.	Input email and password	Able to input text into all text field.	Passed
2.	'Sign Up' button	Able to sign up and redirect to the main page	Passed
3.	Press 'Back' button	Redirect to login page	Passed

Table 6.2.3: Test results for Sign up page

**Main Page Testing**

After successfully login to the application, the system will display the main page to the user which contains the bottom navigation bar and also the connection module page as the first page. Each of the buttons must respond to their respective functionality.

No.	Test Case	Expected Result	Status
1.	Display page after login	Display the first page which is the connection module page	Passed
2.	Press 'Connection' button on bottom navigation bar	Redirect to nearby connection module page	Passed
3.	Press 'Map' button on bottom navigation bar	Redirect to nearby hotspot finder map page	Passed
4.	Press 'Wallet' button on bottom navigation bar	Redirect to payment module page	Passed
5.	Press 'Settings' button on bottom navigation bar	Redirect to settings page	Passed

Table 6.2.4: Test results for Main page



**Connection Module Page Testing**

When the user is on the connection module page, each of the buttons must respond to their respective functionality.

No.	Test Case	Expected Result	Status
1.	'Advertise Hotspot' button	Proceeds to advertise hotspot functionality	Passed
2.	'Discover Hotspot' button	Proceeds to discover hotspot functionality	Passed
3.	'Wi-Fi Settings' button	Redirects to the Wi-Fi Intent	Passed
4.	'Hotspot Settings' button	Redirects to the Hotspot Intent	Passed
5.	Connection info	Display the connection info after a successful connection	Passed

Table 6.2.5: Test results for Connection module page

**Advertise Hotspot Functionality Testing**

Each button in the advertise hotspot functionality must be working and responds to their respective functionality.

No.	Test Case	Expected Result	Status
1.	Starts the Advertising process	Turns on the advertising functionality	Passed
2.	'Stop' Advertising button	Turns off the advertising functionality	Passed
3.	Pop up on connection initiated	Display the incoming connection information from the discoverer	Passed
4.	'Accept Connection' button	Accept the connection from the discoverer	Passed
5.	'Reject Connection' button	Decline the connection from the discoverer	Passed
6.	'Advertiser Confirmation' Page	Display the instruction for the user	Passed

7.	Receive hotspot duration	Able to receive the hotspot duration from the discoverer	Passed
8.	'Open Hotspot Settings' button	Redirects to the Hotspot Intent	Passed
9.	'Send over hotspot information' button	Sends the hotspot credentials to the discoverer	Passed
10.	'Return to homepage' button	Redirects to main page	Passed

Table 6.2.6: Test results for Advertise Hotspot Functionality

### Discover Hotspot Functionality Testing

Each button in the discover hotspot functionality must be working and responds to their respective functionality.

No.	Test Case	Expected Result	Status
1.	Starts the Discovering process	Turns on the discovering functionality	Passed
2.	'Stop' Discovering button	Turns off the discovering functionality	Passed
3.	List of nearby available advertiser	Display the list of nearby available hotspot advertiser	Passed
4.	Tap on specific hotspot advertiser button	Request connection to the hotspot advertiser	Passed
5.	Pop up on connection initiated	Display the outgoing connection information to the advertiser	Passed
6.	'Accept Connection' button	Accept the connection with the advertiser and redirect to main page	Passed
7.	'Reject Connection' button	Decline the connection with the advertiser and redirect to main page	Passed
8.	'Advertiser Confirmation' Page	Display the instruction for the user	Passed
9.	'Send hotspot duration' button	Able to send the hotspot duration to the advertiser	Passed

10.	Receive the hotspot credentials	Able to receive the hotspot credentials from the advertiser	Passed
11.	Display the hotspot credentials	Able to display the hotspot credentials QR code on the screen	Passed
12.	‘Open Wi-Fi Settings’ button	Redirects to the Wi-Fi Intent	Passed
13.	‘Return to homepage’ button	Redirects to main page	Passed

Table 6.2.7: Test results for Discover Hotspot Functionality

### Nearby Hotspot Finder Map Functionality Testing

When the user press on the ‘Map icon’ on the bottom navigation bar, the nearby hotspot finder map page will be redirected and displayed to the user. Each button in the nearby hotspot finder map page must be working and responds to their respective functionality.

No.	Test Case	Expected Result	Status
1.	Fetch data from the database	Fetching the coordinates and timestamp data from the Firebase	Passed
2.	Perform filtering	Filters out irrelevant coordinates and timestamp	Passed
3.	Display hotspot markers	Displaying all the filtered hotspot location by using markers	Passed
4.	On tap of the markers	Display the information of the hotspot in term of timestamp	Passed
		Able to opens up Google Map to navigate to the location	Passed
5.	Tooltip	Display the nearby available hotspot count	Passed

Table 6.2.8: Test results for Nearby Hotspot Finder Map

**Payment Module Functionality Testing**

When the user press on the ‘Wallet icon’ on the bottom navigation bar, the payment module page will be redirected and displayed to the user. Each button in payment module page must be working and responds to their respective functionality.

No.	Test Case	Expected Result	Status
1.	Wallet token balance	Display the total user’s wallet balance	Passed
2.	‘Refresh balance’ button	Refresh and update the latest wallet token balance	Passed
3.	‘Payment to another Account’ button	Perform transactions by transferring wallet token to another user’s wallet	Passed
4.	‘Add Funds’ button	Perform add funds by adding token into the user’s wallet	Passed
5.	‘Remove Funds’ button	Perform remove funds by removing the token from the user’s wallet	Passed
6.	‘Catalogue Page’ button	Redirect to the Catalogue Page	Passed
7.	‘Catalogue Order Voucher’ button	Able to deduct token from the wallet and update the order information to the database	Passed

Table 6.2.9: Test results for Payment Module

**Settings Page Functionality Testing**

When the user press on the ‘Settings icon’ on the bottom navigation bar, the settings page will be redirected and displayed to the user. Each button in the settings page must be working and responds to its respective functionality.

No.	Test Case	Expected Result	Status
1.	Display the profile	Displaying all the profile information including picture and email address	Passed
2.	‘Feedback’ button	Redirect to the Feedback page	Passed

3.	Feedback - Input name, email, and feedback	Able to input text into all text field.	Passed
4.	Feedback – ‘Submit’ button	Able to submit the feedback form to store it in database	Passed
5.	‘FAQ’ button	Redirect to the FAQ page and display the FAQ to the user	Passed
6.	‘Sign Out’ button	Sign out the user from the application and redirect to login page	Passed

Table 6.2.10: Test results for Settings Page

### 6.3 Project Challenges

The following list out the challenges that are being encountered during the development and implementation process of this project:

- **Limitations on accessing Android API.** During the application development process, certain assets must be implemented in order to enable some specific functionalities such as providing network statistics in the application. However, due to the deprecated Android Wi-Fi manager API in Android 10 and above, most of the Wi-Fi manager API is not able to be called hence restricting the ability of the mobile application to provide more functionalities to the user.
- **Technical Limitation of the Nearby Connection API.** The hotspot connection module of this project implements the Nearby Connection API to communicate and exchange the hotspot credentials between both parties. However, the Nearby Connection operates by using a peer-to-peer network connection. The weakness of this peer-to-peer connection is the low range of transmission distance between devices. Hence, devices must stay close to each other in order to advertise and discover each other.
- **Scalability of the smart contract.** The smart contract that is implemented in this project is simple and is not scalable when compared to another complex smart contract. Besides that, there is also a delay issue on the block creation while interacting with the smart contract methods. The block usually takes an

amount of time to create the block on the blockchain network. In addition, as the number of transactions and participants increases, the network can become congested, leading to delays in transaction processing and increased fees.

- **Project timeline management.** One of the project challenges is the difficulty in managing the project timeline. It's challenging to effectively manage the project timeframe. The development process might take longer than expected, hence causing resulting in delays to deliver the final deliverables and also less time to be spent on documentation to make up the gap between the actual and planned times for finishing the code.

### 6.4 Objectives Evaluation

The primary objective of this project is to develop a hotspot-tethering mobile application that provides a platform for mobile data sharing using Flutter. This objective is being met in this project. The application developed in this project allows users to use the Nearby Connection API to advertise and discover nearby hotspots. For users who have excess mobile data, they can use the advertise hotspot functionality to advertise and share their mobile data through sharing hotspot. On the other hand, for users who want to gain Internet access, they can use the discover hotspot functionality to discover and connect to nearby available hotspots.

Next, the application also fulfils the second objective which is to implement smart contracts in the payment system in the application using Solidity. A smart contract is being implemented in this project and is deployed on the Sepolia Test Network. The application can then interact with the smart contract by using the Infura API. Upon successful sharing of a hotspot, the hotspot advertiser will be rewarded with tokens and the hotspot discoverer will transfer the tokens from their wallet to the hotspot advertiser's wallet. This action is performed automatically through the execution of the smart contract function upon successful hotspot-sharing sessions.

## CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION

Another objective which is to facilitates the process of finding Internet access through a nearby hotspot finder using Google Maps API is being achieved in this project. This application uses the Google Maps API to provide a map that display the nearby available hotspot location to the users in the form of markers on the map. The user can then locate and navigate to the nearest hotspot from their location.

Overall, this application has met all of the objectives defined in this project and aims to provide a simple and efficient system for the hotspot sharing activities.

## Chapter 7 Conclusion

### 7.1 Conclusion

In conclusion, the proposed hotspot tethering and sharing mobile application has been successfully implemented. The system is capable of providing a platform for users to share their mobile data. Besides that, the developed mobile application also consists of a nearby hotspot finder functionality that facilitates the process of finding internet access for the users. In addition, a smart contract is also being implemented as the payment system in the application that helps to facilitate the transaction process after the hotspot sharing session. Overall, the system has achieved the initial objectives set at the beginning of the project and is considered to be a success.

In terms of the project's contribution, the hotspot tethering and sharing mobile applications benefits the society by providing cutting-edge features and functions that are hard to find in the present market. The mobile application enables users to instantly share and find internet access. A smart contract is used by this system as the mobile application's payment system, which is worth mentioning. On the blockchain network, the smart contract is being created and implemented, and the mobile application can communicate and interacts with the smart contract for transactions. This system provides a brand-new hotspot tethering and sharing application that works with a smart contract.

Overall, this project has successfully developed a mobile hotspot tethering and sharing application using NFT as well as the smart contract. All of the predefined objectives and project scopes are being achieved. The final deliverables of the project have successfully demonstrated the ability to share and connect to a hotspot, find nearby hotspots, and automate transactions using a smart contract.



### 7.2 Recommendation

This section offers suggestions for further research and improvements to the current system based on the conclusions and learnings from the earlier chapters. The goals of these suggestions are system performance and functionality enhancement for development and expansion.

#### 1. Telco Recommendation System Feature

A telco recommendation system could be integrated into the hotspot tethering and sharing mobile application by providing personalized recommendations for users based on the telco availability, price, and signal strengths in certain areas. This feature seeks to provide recommendations that best suit the customer's mobile data usage and gives better insight to the user upon the telco mobile data subscription.

#### 2. Multi-device connectivity

A mobile hotspot tethering and sharing application with multi-device connectivity can enable users to share their mobile data connection with multiple devices such as smartphones, laptops, tablets, and other devices. This feature can be particularly useful for people who need to work on multiple devices simultaneously or share their internet connection with family or friends.

#### 3. Improved security

Another recommendation is to enhance the security of the hotspot by implementing stronger encryption protocols or incorporating additional authentication methods. This would ensure that only authorized users are able to connect to the hotspot, minimizing the risk of unauthorized access or data breaches.

#### 4. Integration with popular social media platforms

The application could be integrated with popular social media platforms such as Facebook or Twitter, allowing users to share hotspot credentials with their friends and followers with just a few clicks.

### **5. Offline Functionality**

Incorporating local storage capabilities would be necessary to implement an offline mode and allow users to find and locate a nearby hotspot location without an internet connection. Any local updates would synchronize with the Cloud Firestore once the device was connected to the internet.

## REFERENCES

- [1] S. R. D. Statista Research Department, *Number of internet and social media users worldwide as of July 2022*, 20-Sep-2022. [Online]. Available: <https://www.statista.com/statistics/617136/digital-population-worldwide/>. [Accessed: 01-Dec-2022].
- [2] J. Howarth, "Internet traffic from Mobile Devices (2022)," *Exploding Topics*, 30-Nov-2022. [Online]. Available: <https://explodingtopics.com/blog/mobile-internet-traffic>. [Accessed: 01-Dec-2022].
- [3] The World Bank, "Mobile cellular subscriptions (per 100 people)," *The World Bank*. [Online]. Available: [https://data.worldbank.org/indicator/IT.CEL.SETS.P2?end=2020&most\\_recent\\_value\\_desc=true&start=2010&view=chart](https://data.worldbank.org/indicator/IT.CEL.SETS.P2?end=2020&most_recent_value_desc=true&start=2010&view=chart). [Accessed: 01-Dec-2022].
- [4] E. Doku, "Data sink: Over 143 million gigabytes of mobile data unused by customers every month," *Uswitch*. [Online]. Available: <https://www.uswitch.com/media-centre/2018/02/data-sink-143-million-gigabytes-mobile-data-unused-customers-every-month>. [Accessed: 01-Dec-2022].
- [5] M. Constantinescu, E. Onur, Y. Durmus, S. Nikou, M. de Reuver, H. Bouwman, M. Djurica, and P. Maria Glatz, "Mobile tethering: Overview, perspectives and Challengess," *info*, vol. 16, no. 3, pp. 40–53, 2014.
- [6] M. Swan, *Blockchain: Blueprint for a new economy*. O'Reilly, Beijing, 2015.
- [7] J. Frankenfield, "What are smart contracts on the Blockchain," *Investopedia*, 16-Sep-2022. [Online]. Available: <https://www.investopedia.com/terms/s/smart-contracts.asp>. [Accessed: 01-Dec-2022].
- [8] IBM, "What are smart contracts on blockchain?," 01-Oct-2015. [Online]. Available: <https://www.ibm.com/topics/smart-contracts>. [Accessed: 01-Dec-2022].
- [9] June Fabrics Technology Inc. PdaNet+ (Version 5.23) [Mobile app]. Google Play Store. Available: <https://play.google.com/store/apps/details?id=com.pdanet&hl=en&gl=US>
- [10] M. Sauter, "Wi-Fi Direct Connection Establishment," *WirelessMoves*, 23-Jan-2016. [Online]. Available: <https://blog.wirelessmoves.com/2013/01/wi-fi-direct-connection-establishment.html>. [Accessed: 01-Dec-2022].
- [11] Tutorials Point, "Difference between USB tethering and mobile hotspot." [Online]. Available: <https://www.tutorialspoint.com/difference-between-usb-tethering-and-mobile-hotspot#>. [Accessed: 01-Dec-2022].

## REFERENCES

- [12] Mobile Stream. EasyTether Lite (Version 1.1.19) [Mobile app]. Google Play Store. Available:  
<https://play.google.com/store/apps/details?id=com.mstream.e2t&hl=en&gl=US>
- [13] AltheaOrg. Althea Mobile (Version Beta 0 RC11) [Mobile app]. Google Play Store. Available:  
[https://play.google.com/store/apps/details?id=com.althea.althea\\_android&hl=en&gl=US](https://play.google.com/store/apps/details?id=com.althea.althea_android&hl=en&gl=US)
- [14] kirici. Mobile Hotspot (Version 1.0.7) [Mobile app]. Google Play Store. Available:  
<https://play.google.com/store/apps/details?id=com.kirici.mobilehotspot&hl=en&gl=US>
- [15] Simplify Networks. Simplify (Version 5.3.7) [Mobile app]. Google Play Store. Available:  
<https://play.google.com/store/apps/details?id=com.nextwavesimplify&hl=en&gl=US>
- [16] A. Anitha, B. Jitendra, V. Krunal, “Software Process Models for Mobile Application Development: A Review,” International Journal of Computer Science & Communication, 7(1), 150-153, 2

## POSTER

# Mobile Hotspot Tethering and Payment System Using NFT

by Ng Warren Cin Yen



## Introduction

This project propose an mobile hotspot tethering application with payment system integrated. This application allows user to sell their excess cellular data and provide Internet access to the other who are in need.

## Objective

- 1 To develop a hotspot-tethering mobile application that provides a platform for mobile data selling or buying
- 2 To implement smart contracts in the payment system that facilitates transactions.
- 3 To facilitates the process of finding Internet access through a nearby hotspot finder.

## Methodology



Agile SDLC methodology



Developed using Flutter and Solidity



Firebase and Infura API

## Contribution

This application aims to provide decentralized Internet access to the public. It is like Airbnb for Internet access by making Internet more affordable and accessible to everyone. The aim could be achieved by enabling people to share and resell unused or excess Internet access.

## Conclusion

This mobile hotspot tethering and payment system application redefine how the sharing of Internet works. This application will empowers people to resell their access Internet at a more affordable and accessible way to connect more people.