

**DEVELOPMENT OF AN ANDROID-BASE PARKING
MANAGEMENT APPLICATION**

ONG HUI MIN

OPEN UNIVERSITY MALAYSIA

2021

**DEVELOPMENT OF AN ANDROID-BASE PARKING
MANAGEMENT APPLICATION**

ONG HUI MIN

A Final Year Project submitted in fulfilment of the requirements
for the degree of
Bachelor of Information Technology with Honours

Open University Malaysia

2021

DECLARATION

Name: Ong Hui Min

Matric Number: 951126025208001

I hereby declare that this final year project is the result of my own work, except for quotations and summaries which have been duly acknowledged.

Signature:

Date:30/04/2021

DEVELOPMENT OF AN ANDROID-BASED PARKING MANGEMENT APPLICATION

ABSTRACT

Nowadays, available free parking slots are hard to be noticed when the parking area inside the building's parking area is full of cars. When it is hard for a driver to notice available parking, this will result in a longer searching time to find available parking slots. The building installs various sensors in separate parking spaces to detect the cars and sometimes the sensor is not functioning well. Thus, an android-based Parking Management System is developed. This application will be developed to let users enter the building parking area without the parking ticket, display the availability of parking slot numbers so that it will assist the users to easily find available parking slots. Moreover, this application also can let the user pay their parking fee. All the information collected will be sent into the cloud as a database for this system.

PEMBANGUNAN APLIKASI PENGURUSAN KERETA SECARA ANDROID

ABSTRAK

Pada masa kini, slot letak kereta yang kosong adalah sukar untuk diperhatikan apabila kawasan parkir di dalam bangunan yang tempat meletak kenderaan penuh dengan kereta. Pemandu akan ambil masa yang lebih banyak untuk cari satu tempat parkir yang kosong apabila tempat kosong parkir itu sukar untuk dijumpai. Bangunan yang memasang sensor di tempat letak kereta yang berasingan untuk mengesan kereta dan kadangkala sensor tersebut tidak berfungsi dengan baik. Oleh itu, Sistem Pengurusan Kereta telah dikembangkan. Aplikasi ini akan dibangunkan untuk membolehkan pengguna memasuki kawasan parkir bangunan tanpa tiket, menunjukkan ketersediaan nombor slot tempat letak kereta sehingga dapat membantu pengguna untuk mencari slot tempat letak kereta yang tersedia dengan mudah. Selain itu, aplikasi ini juga dapat membiarkan pengguna membayar bayaran letak kereta mereka. Semua maklumat yang dikumpulkan akan dihantar ke cloud sebagai pangkalan data untuk sistem ini.

ACKNOWLEDGEMENT

First, I would like to take an opportunity to thank all the persons who has involved for helping me and assisting me to complete my Final Year Project (FYP). Firstly, I would like to express my deepest appreciation to my project supervisor, Puan Nor Farzana Abd Ghani for her undivided support morally and physically, assistance, guidance, tolerance to complete this project.

Next, I would also like to take the opportunity to thank my family and friends for their patient, understanding and support throughout the process completion of my project. I would also like to thank the people that I met online and gave me solution on helping me to solve my problems. Indeed, throughout the process of doing this project, many problems occurred, but it did not stop me from working hard. In fact, I do all my best to overcome each and every single problem that occurred.

Finally, I would also like to thank for all support and help during my Final Year project.

THANK YOU

ONG HUI MIN

30 Apr, 2021

Table of Contents

1.1	Background to the Study	1
1.2	Problem Statement	1
1.3	Object of the Study	2
1.4	Scope and Limitation	2
1.5	Implementation Plan	3
2.1	Introduction.....	4
2.1.1	Smart Parking Management System	4
2.1.2	Type of Smart Parking Management System	4
3.1	Feasibility Studies.....	7
3.1.1	Cost-Benefit	7
3.1.2	Technical Feasibility	7
3.1.3	Operational Feasibility	7
3.2	Requirement Methods	8
3.2.1	Functional Requirements	8
3.2.2	Non-Functional Requirement	9
3.3	System Development Methods.....	9
3.3.1	Planning Phase	9
3.3.2	Analysis Phase	10
3.3.3	Design Phase	10
3.3.4	Development Phase	10
3.3.5	Testing Phase	11
3.3.6	Deployment Phase.....	11
3.4	Data and Process Modelling Diagrams.....	11
3.4.1	Data Flow Diagram (DFD).....	12
3.4.2	Entity Relation Diagram (ERD).....	13
3.4.3	Unified Modelling Language (UML).....	14
3.5	Database Design.....	15
3.6	Interface Design	17
3.6.1	Registration	17
3.6.2	Login.....	18
3.6.3	ParkNow	19
3.6.4	ExitNow	20
3.6.5	Search Building Parking Slot.....	21
3.6.6	Make Payment.....	22

3.6.7	SOS.....	23
4.1	System Manual	24
4.1.1	Register Page.....	24
4.1.2	Login Page	25
4.1.3	Home Page.....	26
4.1.4	ParkNow	27
4.1.5	ExitNow	28
4.1.6	Building	29
4.1.7	User Parking Detail	33
4.1.8	User.....	35
4.1.9	SOS.....	36
4.2	Testing Plan and Testing Output	37
4.3	Main Function Codes	41
4.3.1	User Registration.....	41
4.3.2	User Login	43
4.3.3	ParkNow	43
4.3.4	ExitNow	44
4.3.5	Building Parking Dashboard.....	45
4.3.6	Building Report Parking.....	47
4.3.7	User Parking Detail	48
4.3.8	SOS.....	49
5.1	Summary.....	50
5.2	Limitations of The System	50
5.3	Future Development.....	50
5.3.1	Empty slot booking	50
5.3.2	Location reminder	50

LIST OF TABLES

Table 1.1: Gantt Chart for first semester

Table 1.2: Gantt Chart for second semester

Table 3.1: Functional Requirement Table

Table 3.2: Non-Functional Requirement Table

Table 3.3: Device Requirement

Table 4.1: Table of User Registration Testing Plan and Output

Table 4.2: Table of User Login Testing Plan and Output

Table 4.3: Table of User Detail Testing Plan and Output

Table 4.4: Table of ParkNow Testing Plan and Output

Table 4.5: Table of ExitNow Testing Plan and Output

Table 4.6: Table of Building Parking Dashboard Testing Plan and Output

Table 4.7: Table of Building Report Parking Testing Plan and Output

Table 4.8: Table of User Parking Detail Testing Plan and Output

Table 4.9: Table of SOS Testing Plan and Output

LIST OF FIGURES

Figure 2.1: Parking Space Available according to the level

Figure 2.2: LED Sensor Detection

Figure 2.3: Online Reservation System

Figure 3.1: show the DFD of User Registration

Figure 3.2: show the DFD of User Login

Figure 3.3: show the DFD of User Enter Building

Figure 3.4: show the DFD of User Search Parking Slot

Figure 3.5: show the DFD of User Make Payment

Figure 3.6: shows the ERD of the Parking Management Application

Figure 3.7: shows the UML of the Parking Management Application

Figure 3.8: Parking Management Application firebase table structure

Figure 3.9: User firebase table structure and data

Figure 3.10: Building table structure and data

Figure 3.11: Building Parking Slot table structure and data

Figure 3.12: Register Page

Figure 3.13: Login Page

Figure 3.14: ParkNow Page

Figure 3.15: ExitNow Page

Figure 3.16: Search Parking Slot

Figure 3.17: Payment method

Figure 3.18: SOS Page

Figure 4.1: Register Page

Figure 4.2: Login

Figure 4.3: Home Page

Figure 4.4: ParkNow Page

Figure 4.5: ExitNow Page

Figure 4.6: Building Page

Figure 4.7: Parking Dashboard – select level

Figure 4.8: Parking Dashboard – parking slot list

Figure 4.9: Report Page

Figure 4.10: User Parking Detail Page

Figure 4.11: Payment Method Page

Figure 4.12: User Profile Page

Figure 4.13: SOS Page

Figure 4.14: Check user attribute code

Figure 4.15: Check user existed code

Figure 4.16: Add new user

Figure 4.17: Check user existed code

Figure 4.18: Check building existed code

Figure 4.19: Update user parking detail code

Figure 4.20: Check building existed code

Figure 4.21: Update user parking detail

Figure 4.22: Get building level list code

Figure 4.23: Get parking slot according to selected level

Figure 4.24: Validate all attribute code

Figure 4.25: Validate parking slot code

Figure 4.26: Get user parking detail code

Figure 4.27: Calculate total parking fee amount

Figure 4.28: Phone call code

LIST OF ABBREVIATION

PGIS	Parking Guidance and Information System
LED	Light Emitting Diode
SDLC	System Development Life Cycle
DFD	Data Flow Diagram
ERD	Entity Relation Diagram
UML	Unified Modelling Language
IDE	Integrated Development Environment
QR Code	Quick Response Code

CHAPTER 1

INTRODUCTION

1.1 Background to the Study

As the population increased in the metropolitan cities, the usage of vehicles got increased. Finding a car parking space in most metropolitan areas, especially during rush hours. For example, shopping mall. Some of the people they are not familiar with the shopping mall parking area because the parking area has many lanes or slots for car parking is difficult for drivers. Parking violations may cause damage to the car. Thus, there is a need to provide sufficient parking spaces coupled with plenty of slots to help the user park his car safely. Basically, the parking system is one of the most adopted and fastest-growing solutions to a smart city. Currently, most of the existing car parks do not have a systematic system.

Every user's demands should be users friendly, should be more efficient, they should provide more security. The idea of Parking Management Application is to help the user easy to find a parking slot. In this application, the user can view various parking areas and select the parking area to view whether a parking lot is available or not. Once the user is parked the car they can scan the parking slot QR code to record it. The user not need to keep their parking ticket because Parking Management Application will provide a QR code for user while they want to parking in a shopping mall or parking building. Since now the government promotes cashless transactions, therefore, this application also will provide online make payments. The user can use this application to report an emergency when they were in a danger.

1.2 Problem Statement

Hard to find a parking slot this problem is everyone that will face especially if we are going to travel by car and go to shopping mall that we are not familiar with their parking area, during rush hour finding an available parking slot is even more difficult. Every day thousands of car drivers spend a lot of time and fuel is wasted to find where to park. The result of this situation will cause some irresponsible people parking violations for example they will park at the parking slot that provide for people with limited mobility, lady parking slot and other. This action will bring a lot of inconvenience for other people. The purpose of lady parking slot is to reduce the risk of lady encountering therefore is some people is abusing this parking slot lady facing danger chances will be increasing. Moreover, there are some people who will

accidentally lose their parking ticket will cause them need to pay a penalty of lost their parking ticket.

1.3 Object of the Study

The main objective of this parking management application is:

- To make the driver easily to find an available parking slot and no need to spend a lot of time and fuel is wasted to find where to park.
- To reduce the irresponsible people parking violations because this application can let user report if they found out there are parking violation.
- To let a user easy to make payment for their parking ticket, thus, they no need to queue to make payment using a machine for example e-wallet, online payment.
- To reduce user's chances of facing danger.
- If a user-facing an emergency problem this application immediately provides help for example when the user presses the SOS button to notify the police station that nearest the user and emergency contact that user set.
- Help users avoid unnecessary overhead for example lost their parking tickets will cause them to need to pay a penalty of lost their parking ticket.

1.4 Scope and Limitation

This project covered:

- This parking management application will show which area has an available parking slot.
- This application will let the user set their emergency contact person once the user presses the SOS button will contact the police station near that user and it will also contact the emergency contact person that the user set.
- This application provides a report function that will let the user report about there are some irresponsible people parking violations.
- This application also lets the user manage their parking ticket user can view their parking ticket detail like parking time, parking charges, parking slot number and other.

1.5 Implementation Plan

Table 1.1: Gantt Chart for first semester

No	Task Name	1	2	3	Week 4	5	6	7	8	9	10
1	Chapter 1 : Introduction										
1.1	Background										
1.2	Problem Statement										
1.3	Objectives										
1.4	Scope and Limitation										
1.5	Implementation Plan										
2	Chapter 2 : Literature Review										
2.1	Research about Parking Management System										
3	Chapter 3 : System Analysis and Design										
3.1	Feasibility Studies										
3.2	Requirement Methods										
3.3	System Development Methods										
3.4	Data And Process Modelling Diagrams										
3.5	Program Design, Database Design, Interface Design										

Table 1.2: Gantt Chart for second semester

No	Task Name	1	2	3	Week 4	5	6	7	8	9	10
1	Chapter 4 : System Implementation and Testing										
1.1	System Guides / Manual										
1.2	Installation Manual										
1.3	Testing Plan, Test Output										
1.4	Scope and Limitation										
1.5	Main Function Codes										
2	Summary and Conclusion										
2.1	Summary Of Main Findings										
2.2	Discussion And Implications										
2.3	Limitations Of The System										
2.4	Future Development										
3	REFERENCES										
4	APPENDICES										

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter is to investigate and study the current practice and recent research that related to mobile application parking management application implementation. Sources that will be reviewed included research papers that published in professional journal or website and other. This chapter will also summarize and discuss about all the papers studied.

2.1.1 Smart Parking Management System

Parking facilities have always been important because they allow drivers to safely leave their cars while performing daily activities. Generally, the information provided by the smart parking system and the guidance implemented is very useful in assisting the driver in finding available space. With the implementation of new technologies, the payment of parking fees has become easier. Sensors are used to help detect the presence of cars. This is absolutely necessary when developing a smart parking management system because information about parked vehicles is needed. The information can be easily collected from the sensor so that the system can use it, and the same information will also be sent to the driver.

2.1.2 Type of Smart Parking Management System

The automated parking system can mostly be divided into four different types which are Parking Guidance and Information System (PGIS), Electronic Parking, Image-Based Parking, and Street Parking System. Different types have different functionalities of which they address various issues related to the car parking facilities. The below section will explain the implementation and characteristics mixed with examples that can be found around the world.

2.1.2.1 Parking Guidance and Information System (PGIS)

According to Iyaka Beni, Parking Guidance and Information System (PGIS) can be categories into two different features. This can be implemented to monitor the entire building or just specific parking (Iyaka Beni, n.d.). The mentioned categories are mostly implemented in big parking spaces. Parking Guidance and Information System (PGIS) offers basically the same

benefit as the Smart Parking System which was mentioned previously. The major similarity comes in decision-making. The information which is provided by the system helps the drivers to make decisions on how to reach their intended destinations and also in locating available parking space in the parking facility. In Parking Guidance and Information System (PGIS) used variable message signs to provide drivers with ways to take when looking for an empty spot. Figure 2.1 show the number of available parking slot according to the level.



Figure 2.1: Parking Space Available according to the level

Figure 2.2 shows the placement of various sensors in separate parking spaces to detect the presence of cars. LED lights and sensors are linked together, and then the sensors are placed in each parking space in the parking lot. This is used to indicate that a parking space has been occupied. Then, according to the different sensors, before outputting the available light spots, the occupancy of each light spot is counted. From the information obtained, the execution process will analyse the information to display accurate information for the driver to view.

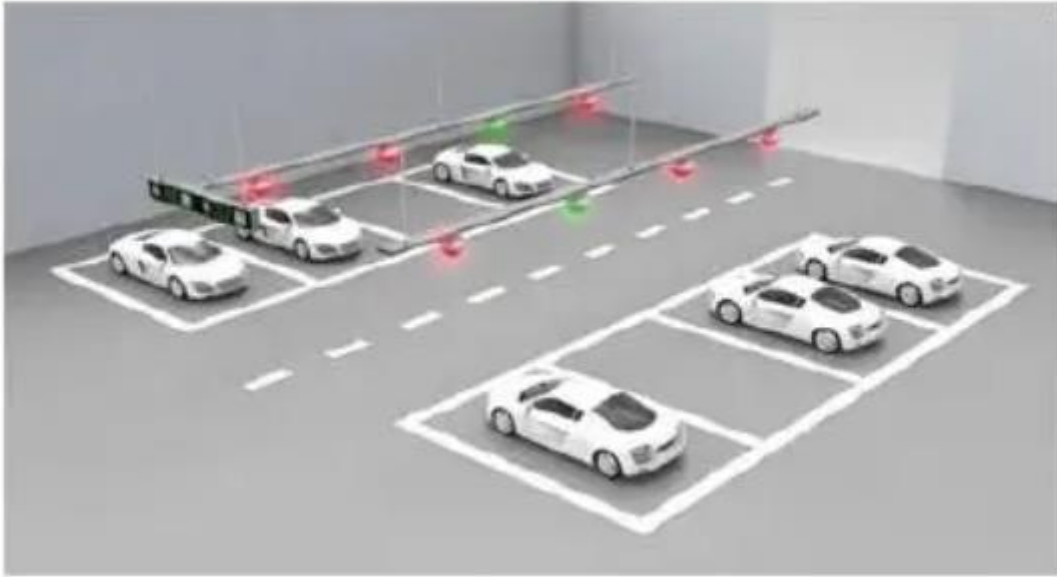


Figure 2.2: LED Sensor Detection (Iyaka Beni, n.d.)

2.1.2.2 Electronic Parking

According to Iyaka Beni, this is a parking system that allows the driver to select or request the availability of parking spaces. If there are vacancies, they can reserve a parking space to ensure that there will be no parking problems when they arrive at their destination. The electronic parking system allows drivers to reserve parking spaces through various methods such as telephone and online. The advantage of using an electronic parking system is that the driver does not need to find all the trouble of finding a parking space. (Iyaka Beni, n.d.).



Figure 2.3: Online Reservation System (Iyaka Beni, n.d.)

CHAPTER 3

SYSTEM ANALYSIS AND DESIGN

3.1 Feasibility Studies

3.1.1 Cost-Benefit

Cost benefit attempts to weigh the cost of developing and implementing the new system and the benefits of installing the new system in place. This feasibility study provides senior management with an economic basis for the new system. In this case, a simple economic analysis that actually compares costs and benefits is more meaningful. In addition, this proved to be a useful reference for comparing actual costs during the course of the project. Automation may bring various intangible benefits. These measures may include improving customer satisfaction, improving operational accuracy, improving document and record retention, and speeding up information retrieval.

3.1.2 Technical Feasibility

Technical feasibility is concentrated on existing systems, such as hardware, software, etc. And to what extent can support the proposed increase. For example, if the current computer is running at 80% capacity, running other applications may overload the system or require additional hardware. This involves financial considerations to accommodate technical improvements. If the budget is severely insufficient, the project can be judged, but it is not feasible.

3.1.3 Operational Feasibility

The proposed project will only be beneficial if it is transformed into an information system that meets the operational requirements of the organization. In short, the feasibility test will ask whether the system can run during development and installation. What are the main obstacles to implementation? The following are questions that help test the feasibility of project operations.

3.2 Requirement Methods

The requirement method can be divided into two type which is functional requirement and non-functional requirement. Functional mean that it will providing particular service to the user.

3.2.1 Functional Requirements

These are the type of requirements that defines the functionalities of the system. Looking at the parking management application, the parking should be automated. Automated which means including artificial intelligence, should perform most of the tasks without the aid of the human being. These include using an ultrasonic sensor to recording the status of the parking slot. Allow the user to search the parking slot and not need to drive the car to go around to find an available parking slot and the user can make a payment through their phone without using the parking ticket paying machine.

Table 3.1: Functional Requirement Table

No.	Functional Requirement
1	Develop register new user function
2	Develop login function
3	Develop enter parking building parking in other to calculate the parking fee.
4	Develop exit parking building function
5	Develop record user parking slot function which can let users can the parking slot QR code
6	Develop pay parking fee function
7	Develop SOS function
8	Develop report illegal parking function
9	Send data to firebase to create a new data or update the data

3.2.2 Non-Functional Requirement

These requirements describe various aspects that can be used to analyse the operation of the system. From the perspective of the parking management application, the functions added to the parking management application should be clear and easy to use, so that users will not waste time trying to use the system. Unnecessary functions should be avoided so that the system does not slow down when processing data. Functions such as searching for parking spaces should be easy to use and effective to minimize data processing speed.

Table 3.2: Non-Functional Requirement Table

No.	Non-Functional Requirement
1	No delay in the application
2	Fast data processing
3	Simple search available parking slot process
4	Multiple online payment type can be chosen
5	Simple and no complexity interface design
6	Easy to use

3.3 System Development Methods

In this section, the flow of the project will discuss briefly to give more understanding of the design and development of this project. There are many methods that can be used for developing this project. The most suitable methodology method to implement in this project is System Development Life Cycle (SDLC). This methodology is based on phases for each development process. Every phase of this methodology will be explained.

3.3.1 Planning Phase

In this phase will have a brainstorming idea that what problem needs to be solved. The purpose of these projects is constructed based on what we want to be achieved. We determined the problem that we need to be solved. Determine the system scope based on the system scope,

user scope, and functional scope to identify the important functions that need to be included in the system in order to achieve the planned goals.

3.3.2 Analysis Phase

In Analysis phase is to analyse the existing Parking Management Application and all function and requirement that is needed to design and develop a new application. In this phase, collected and gathered information through a journal, articles, books, and research papers that is regarding the Parking Management Application that using android devices. And based on the information collected, the method, and technique that is suitable for this project had been decided.

3.3.3 Design Phase

According to the functionality that wants to be built to define the design of the application and developed a prototype. The structural diagram to show the flow of the application is Data Flow Diagram (DFD), Entity Relation Diagram (ERD), and Unified Modelling Language (UML). All these structural diagrams are helping in developing the application.

3.3.4 Development Phase

To develop the application, the software I use is Android Studio which is an official Integrated Development Environment (IDE) for android application development software. The programming language that will use is Java. To manage the data that will show to the user and to store the data that will receive by the user is Firebase which will manage all data in real-time in the database. So, the exchange of data to and from the database is easy and quick.

For the application to run on an android smartphone, the device is expected to meet the following system requirements.

Table 3.3: Device Requirement

Operating system	At least Android 5.1 (Lollipop)
Processor	Intel Atom® Processor Z2520 1.2 GHz, or faster processor
RAM	Minimum 2GB
Connection	Wi-Fi and 4G (data services) to be able to work with data services.
Camera	Yes
Location	GPS

3.3.5 Testing Phase

In this phase, the application testing is to ensure all the module functions well. Any error or bug that occurs will be fixed and repeated testing the application until all the modules function well.

3.3.6 Deployment Phase

In this phase, when the application fully functions well and it is time to deployed the system and let the user use the application. Once the application is in a stable state, it will check whether the system meets all the goals.

3.4 Data and Process Modelling Diagrams

This section will review the design phase in project development. Design is the process of the final product and its proposed thinking framework. It represents a model of how to reach a particular goal in a project. Whereas this model represents a set of strategies to achieve design goals. Before developing the application, the modelling process involves the processes of application development. In this project, Data Flow Diagram (DFD), Entity Relationship

Diagram (ERD), and Unified Modelling Language (UML) were come out first before the development of the system, so that the system can develop successfully.

3.4.1 Data Flow Diagram (DFD)

The figure shows the flow in this system starting with the user registration, login, entry into the building with QR code, exits the building with QR code, search building parking slot, and make payment.



Figure 3.1: show the DFD of User Registration

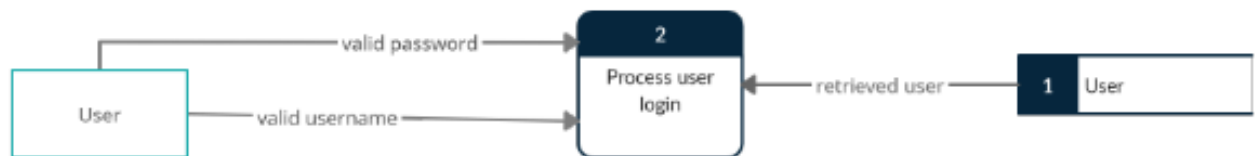


Figure 3.2: show the DFD of User Login



Figure 3.3: show the DFD of User Enter Building

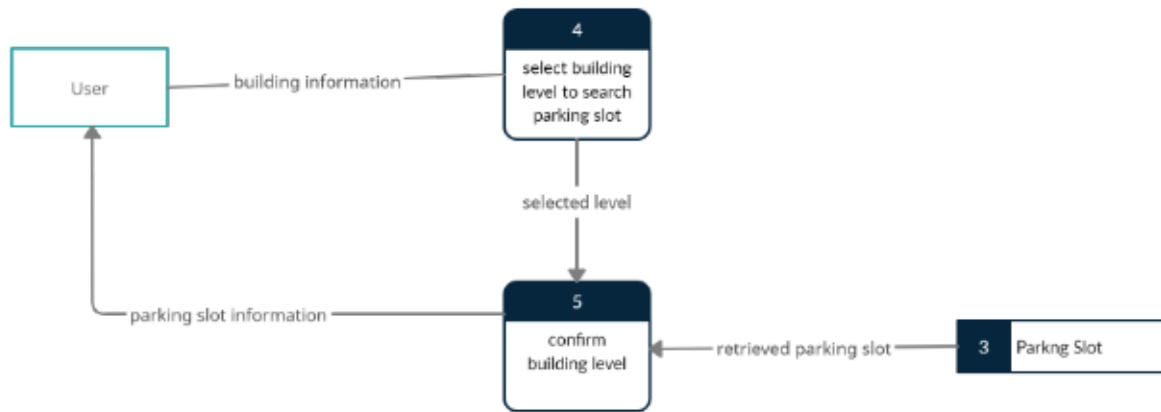


Figure 3.4: show the DFD of User Search Parking Slot

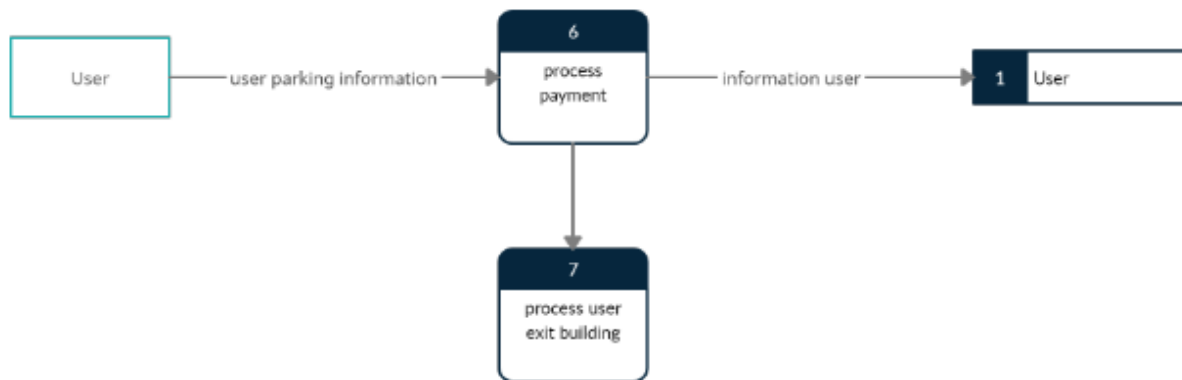


Figure 3.5: show the DFD of User Make Payment

3.4.2 Entity Relation Diagram (ERD)

An Entity Relationship Diagram (ERD) is a flow chart that illustrates how "entities" such as people, objects, or concepts in the system are related to each other. ERD is the most commonly used diagram for designing or debugging relational databases in the fields of software engineering, business information systems, education, and research.

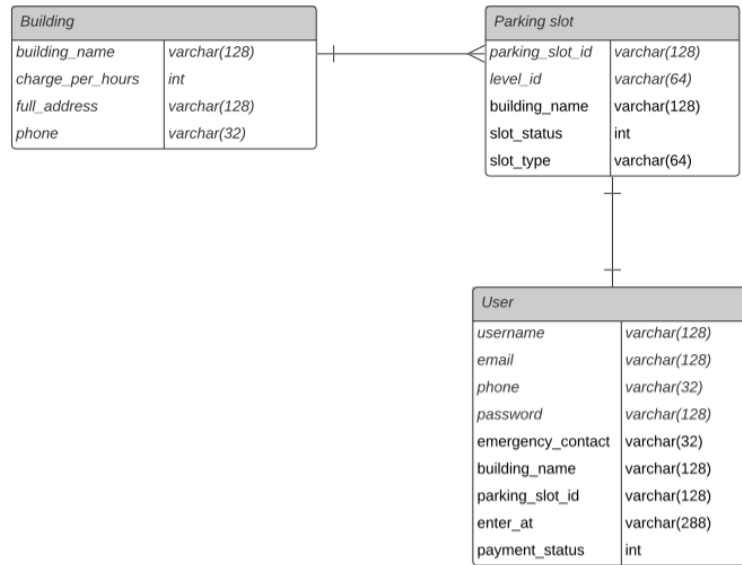


Figure 3.6: shows the ERD of the Parking Management Application.

3.4.3 Unified Modelling Language (UML)

Unified Modelling Language (UML) is a method to visually express the architecture, design, and implementation of complex software systems. When writing code, there are thousands of lines in the application, so it is difficult to track the relationships and hierarchies in the software system. The UML diagram divides the software system into components and subcomponents.

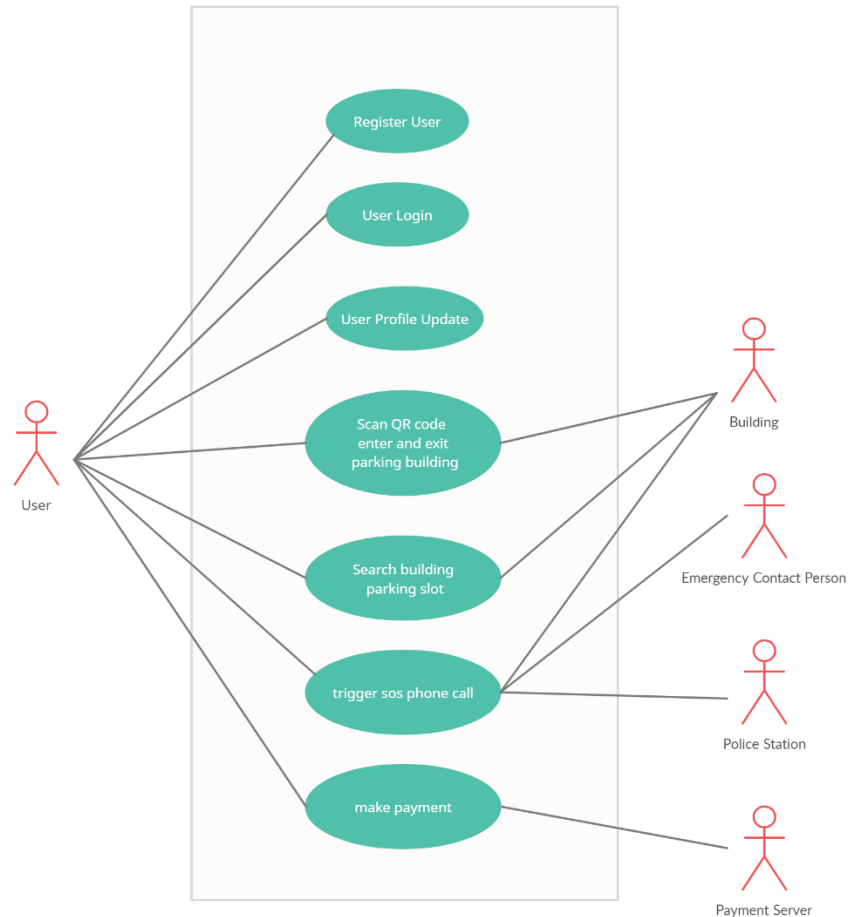


Figure 3.7: shows the UML of the Parking Management Application.

3.5 Database Design

The database is the data collection of the system. They support electronic storage and data processing. The database makes data easy to manage, accessed, and updated. It is usually regarded as a collection of records and each record has its own structure and input data type. The list of tables involved in the system is as follows:

Figure 3.1 show all tables used in system including user, building, and parking slot.

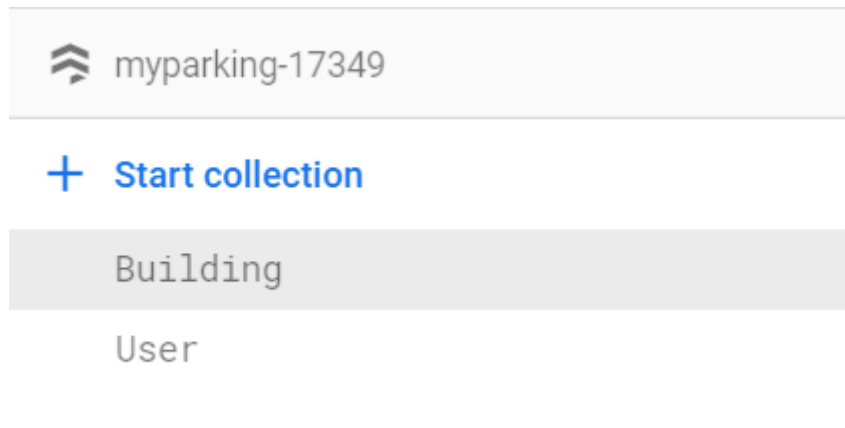


Figure 3.8: Parking Management Application firebase table structure.

<div> <div> <div></div> <div>myparking-17349</div> </div> <div> <div>+ Start collection</div> <div>Building</div> <div>User ></div> </div> </div>	<div> <div> <div></div> <div>User</div> </div> <div> <div>+ Add document</div> <div>MeiLing ></div> <div>adam</div> <div>ali</div> </div> </div>	<div> <div> <div></div> <div>MeiLing</div> </div> <div> <div>+ Start collection</div> <div>+ Add field</div> <div>building_name: "</div> <div>email: "meiLing@gmail.com"</div> <div>emergency_contact: "0104415269"</div> <div>enter_at: "0"</div> <div>paid_price: 0</div> <div>parking_slot_id: "</div> <div>password: "abc123"</div> <div>payment_status: 0</div> <div>phone: "01144125225"</div> <div>username: "meiLing"</div> </div> </div>
--	---	---

Figure 3.9: User firebase table structure and data.

<div> <div> <div></div> <div>myparking-17349</div> </div> <div> <div>+ Start collection</div> <div>Building ></div> <div>User</div> </div> </div>	<div> <div> <div></div> <div>Building</div> </div> <div> <div>+ Add document</div> <div>Alor Star Mall ></div> <div>Aman Central</div> </div> </div>	<div> <div> <div></div> <div>Alor Star Mall</div> </div> <div> <div>+ Start collection</div> <div>L1</div> <div>L2</div> <div>+ Add field</div> <div>building_name: "Alor Star Mall"</div> <div>charge_per_hours: 1</div> <div>full_address: "Alor Star Mall, 05400 Alor Setar, Kedah"</div> <div>parking_level <div>0 "L1"</div> <div>1 "L2"</div> </div> <div>phone: "047569825"</div> </div> </div>
--	---	--

Figure 3.10: Building table structure and data

Alor Star Mall	L1	L1-ZA-C1
<div>+ Start collection</div> <div>L1 ></div> <div>L2</div> <div>+ Add field</div> <div> building_name: "Alor Star Mall" charge_per_hours: 1 full_address: "Alor Star Mall, 05400 Alor Setar, Kedah" parking_level <div> <div>0 "L1"</div> <div>1 "L2"</div> </div> phone: "047569825" </div>	<div>+ Add document</div> <div>L1-ZA-C1 ></div> <div> L1-ZA-C2 L1-ZA-C3 L1-ZA-C4 L1-ZB-C1 L1-ZB-C2 L1-ZB-C3 </div>	<div>+ Start collection</div> <div>+ Add field</div> <div> building_name: "Alor Star Mall" level_id: "L1" slot_id: "L1-ZA-C1" slot_status: 0 slot_type: "Lady" </div>

Figure 3.11: Building Parking Slot table structure and data

3.6 Interface Design

3.6.1 Registration

Register

Username

Email

Phone Number

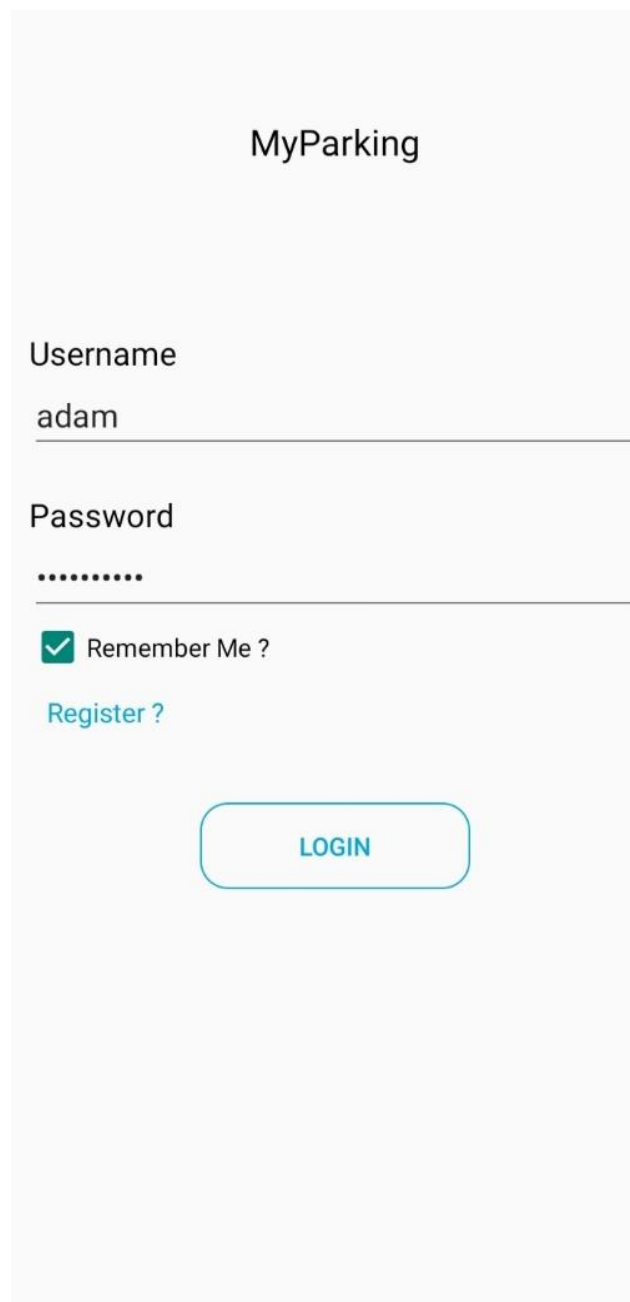
Password

Confirm Password

REGISTER

Figure 3.12: Register Page

3.6.2 Login



The image shows a login page for 'MyParking'. It features a light gray background with a white login form in the center. The form has a title 'MyParking' at the top. Below it are two input fields: 'Username' with the text 'adam' and 'Password' with masked characters '.....'. There is a checked checkbox labeled 'Remember Me ?' and a link 'Register ?' in blue. At the bottom of the form is a blue 'LOGIN' button.

MyParking

Username
adam

Password
.....

☒ Remember Me ?

[Register ?](#)

LOGIN

Figure 3.13: Login Page

3.6.3 ParkNow

ParkNow



Please Scan Building QR Code

Figure 3.14: ParkNow Page

3.6.4 ExitNow

ExitNow



Please Scan Building QR Code

Figure 3.15: ExitNow Page

3.6.5 Search Building Parking Slot

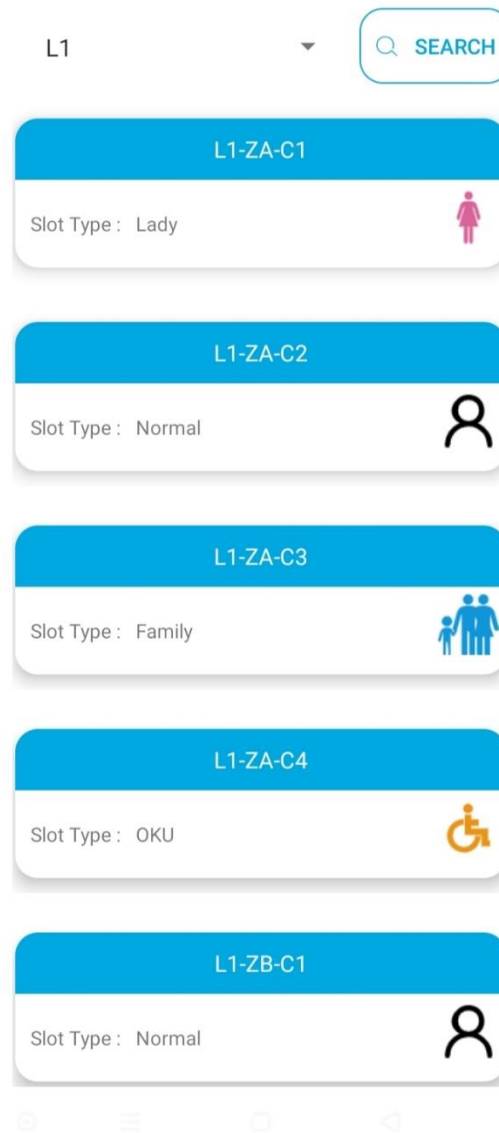
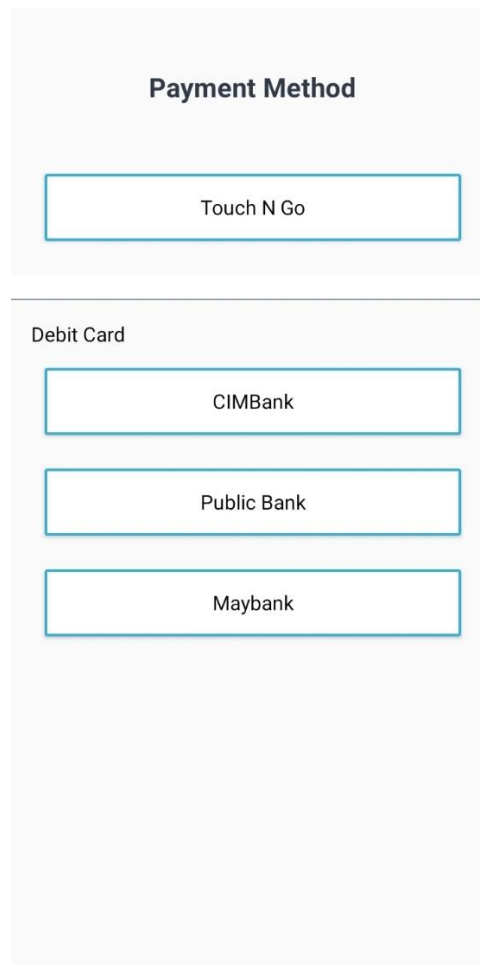


Figure 3.16: Search Parking Slot

3.6.6 Make Payment



The image shows a mobile application interface for selecting a payment method. At the top, there is a section titled "Payment Method" with a light gray background. Below this title is a single button labeled "Touch N Go". A horizontal line separates this section from the one below. The second section is titled "Debit Card" and contains three buttons stacked vertically, labeled "CIMBank", "Public Bank", and "Maybank". All buttons have a light gray background and a thin blue border.

Payment Method

Touch N Go

Debit Card

CIMBank

Public Bank

Maybank

Figure 3.17: Payment method

3.6.7 SOS

SOS

Malaysia Emergency Response Services

999

Building Name :

Alor Star Mall

Phone :

047569825

Emergency Contact

0107763310

Figure 3.18: SOS Page

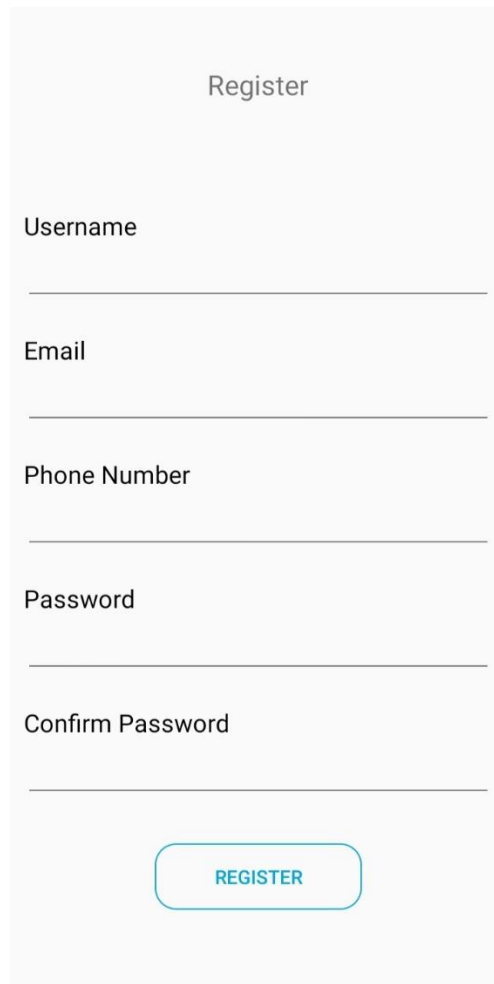
CHAPTER 4

SYSTEM IMPLEMENTATION AND TESTING

4.1 System Manual

4.1.1 Register Page

In register page user need to fill in all the attribute. The password format must be more than 8 character, contain at least 1 digit, 1 alphabet, 1 special symbol. The password and confirm password must be same otherwise a warning message will show.

A screenshot of a web registration form titled "Register". The form is set against a light gray background. It contains five input fields, each with a label to its left: "Username", "Email", "Phone Number", "Password", and "Confirm Password". Each label is followed by a horizontal line representing the input field. At the bottom of the form, there is a blue rounded rectangular button with the word "REGISTER" in white capital letters.

Register

Username

Email

Phone Number

Password

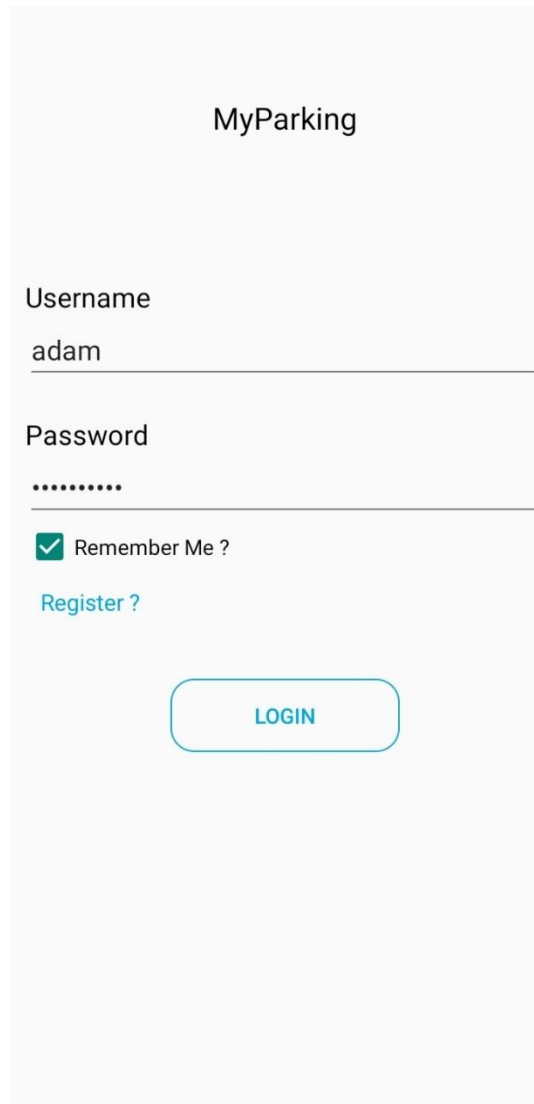
Confirm Password

REGISTER

Figure 4.1: Register Page

4.1.2 Login Page

In login page, user must fill in the username and password. The purpose of remember me is to let user not need to fill in their username and password when they check this check box in their first-time login.



MyParking

Username
adam

Password
.....

☒ Remember Me ?
[Register ?](#)

LOGIN

Figure 4.2: Login Page

4.1.3 Home Page

After user success login it will redirect user to Home Page. In this page user can choose the function according their need.

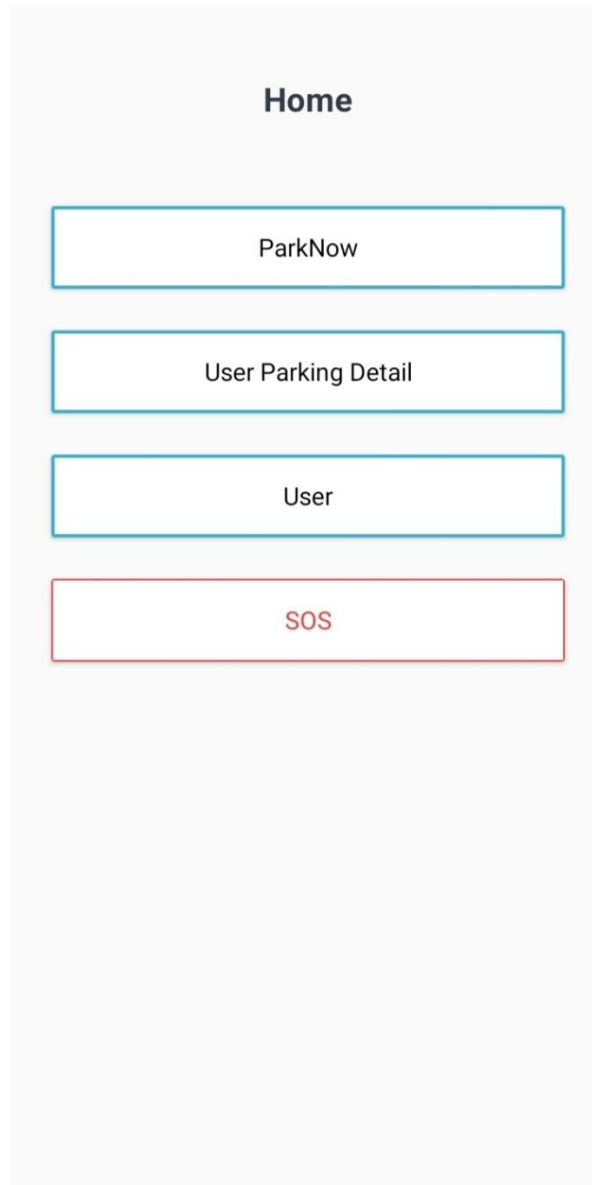


Figure 4.3: Home Page

4.1.4 ParkNow

If user need to enter a building to parking it will require user to scan building QR code, therefore user can choose ParkNow in the Home page and it will redirect user to this page. Click scan icon to open the camera to scan the building QR Code.

ParkNow



Please Scan Building QR Code

Figure 4.4: ParkNow Page

4.1.5 ExitNow

If the user wants to exit the building, it will require user to scan building QR code, therefore user can choose ExitNow in the Home page and it will redirect user to this page. Click scan icon to open the camera to scan the building QR Code. If the user didn't make the payment a warning message will show to user.

ExitNow



Please Scan Building QR Code

Figure 4.5 ExitNow Page

4.1.6 Building

If user need to know more about this building. User can choose the card that show by building name and it will redirect user to this page. In this will let user choose building info, parking dashboard and report.

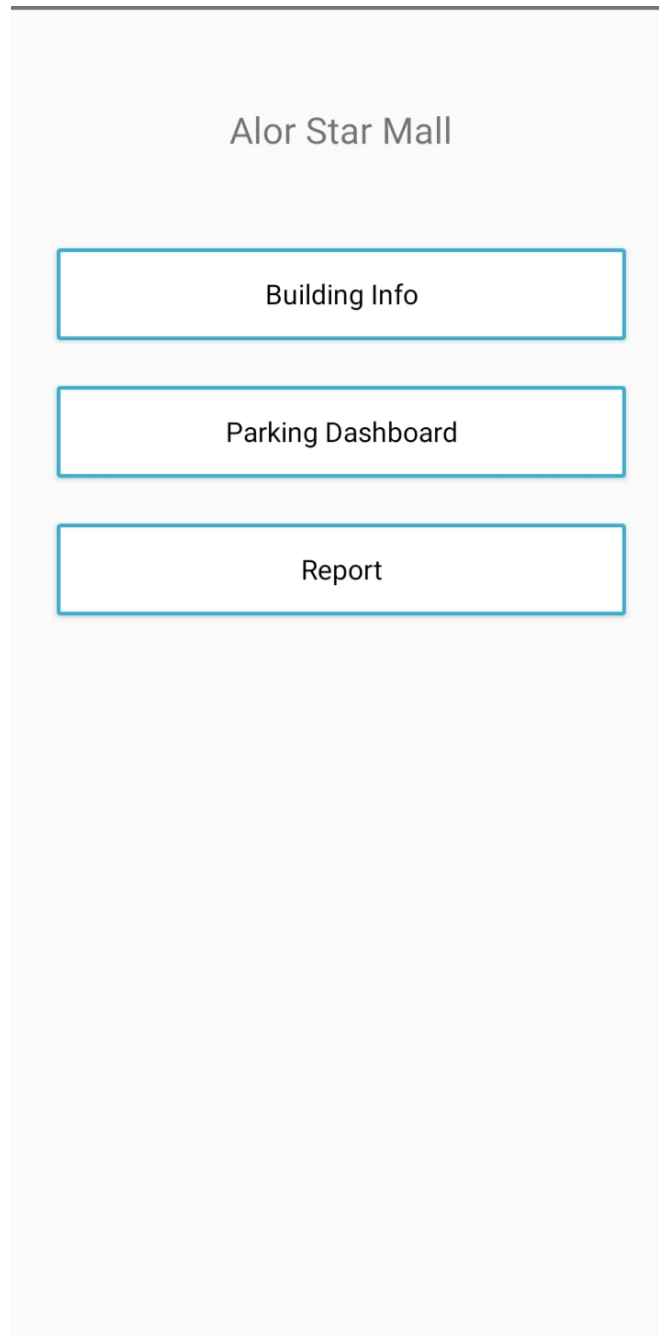
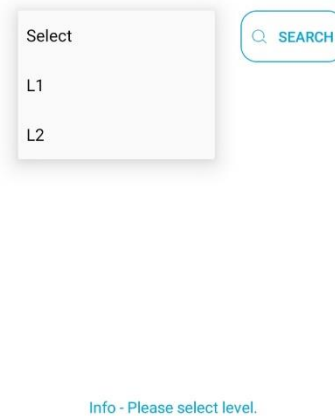


Figure 4.6: Building Page

4.1.6.1 Parking Dashboard

If user want to search the available parking slot, user can choose the Parking Dashboard in the Building page. And it will redirect user to this page. To view the available parking slot, first, user need to choose level which is show in figure 4.7. Next, user need to press search button to get all the available parking slot list that belong this level which will show in figure 4.8. Each of the parking slot will show the type of the parking slot.



The image shows a user interface for selecting a parking level. On the left, there is a dropdown menu with a light gray background. The word "Select" is at the top in a small, dark font. Below it, the options "L1" and "L2" are listed in a slightly larger, dark font. To the right of the dropdown is a rounded rectangular button with a teal border and a teal background. It contains a magnifying glass icon followed by the word "SEARCH" in teal capital letters. Below these elements, centered, is a small teal text message: "Info - Please select level."

Figure 4.7: Parking Dashboard – select level

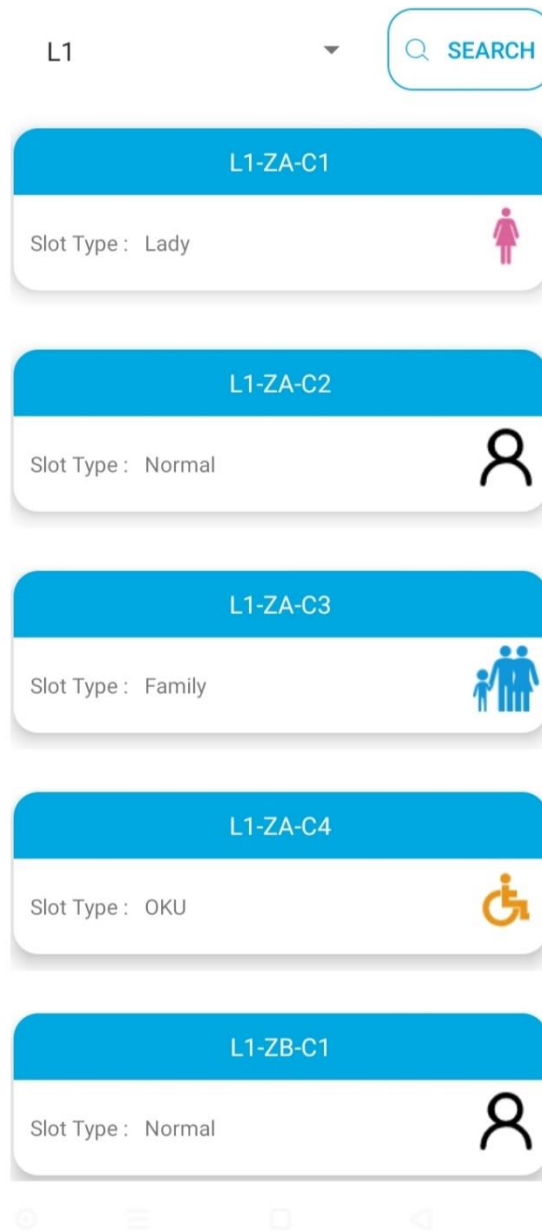
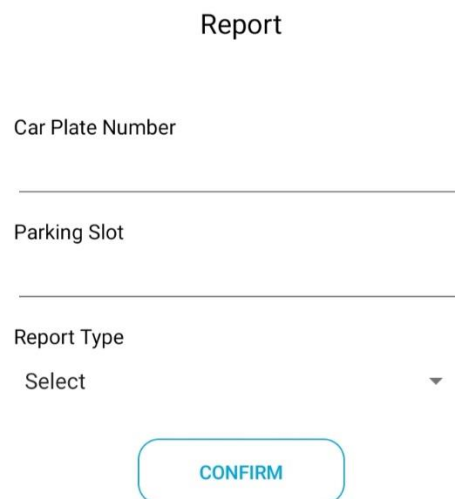


Figure 4.8: Parking Dashboard – parking slot list

4.1.6.2 Report

If user found out there are illegal parking which is happen in the building parking space. User can click the Report card which is show in Building page. In this page user must fill in the car plate number, parking slot, and select the report type otherwise a warning message will show to user.



The image shows a web form titled "Report". It contains three input fields: "Car Plate Number", "Parking Slot", and "Report Type". The "Report Type" field is a dropdown menu with "Select" as the current selection. Below the fields is a blue "CONFIRM" button.

Report

Car Plate Number

Parking Slot

Report Type

Select

CONFIRM

Figure 4.9: Report Page

4.1.7 User Parking Detail

If user want to record parking slot, view current total parking fee that had been charge and make payment. In Home page user can choose User Parking Detail and it will redirect user go to this page figure 4.10. To record the parking slot user can click scan icon to scan the parking slot QR code. To make the payment user can click the pay button after click the pay button it will redirect user to Payment Method page which will show in figure 4.11. In this will show multiple payment method for user choose and once the payment is completed the it redirects user back to this page and the pay button will hide.

Parking Detail

Building Name :

Alor Star Mall

Parking Slot

Enter Time

2021-05-02 19:29:39

Total Amount

PAY

Figure 4.10: User Parking Detail Page

Payment Method

Touch N Go

Debit Card

CIMBank

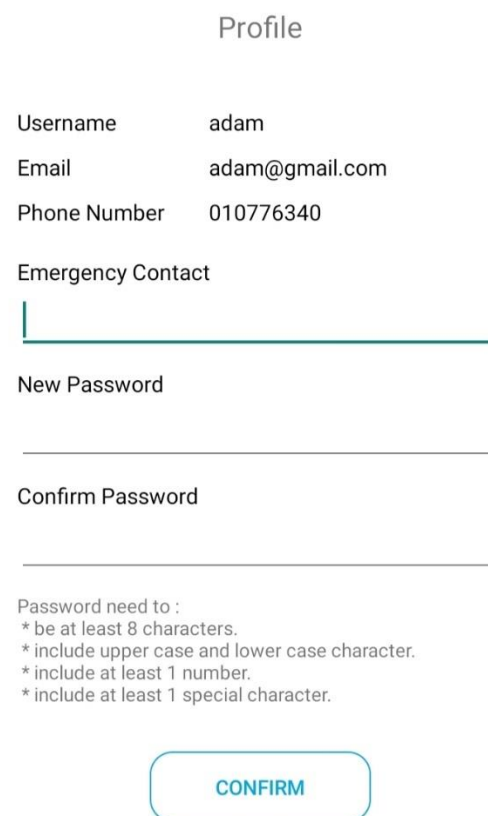
Public Bank

Maybank

Figure 4.11: Payment Method Page

4.1.8 User

If user want to update their information. User can choose User card in the Home page and it will redirect user to this page which will show in figure 4.12. In this user can update their emergency contact and password. The password format same as register must be more than 8 character, contain at least 1 digit, 1 alphabet, 1 special symbol. The password and confirm password must be same otherwise a warning message will show.



The form is titled "Profile" and contains several input fields and a confirmation button. The fields are: Username (filled with "adam"), Email (filled with "adam@gmail.com"), Phone Number (filled with "010776340"), Emergency Contact (empty), New Password (empty), and Confirm Password (empty). Below the password fields, there are four bullet points listing password requirements: "Password need to :", "* be at least 8 characters.", "* include upper case and lower case character.", "* include at least 1 number.", and "* include at least 1 special character.". At the bottom of the form is a blue "CONFIRM" button.

Username	adam
Email	adam@gmail.com
Phone Number	010776340
Emergency Contact	
New Password	
Confirm Password	

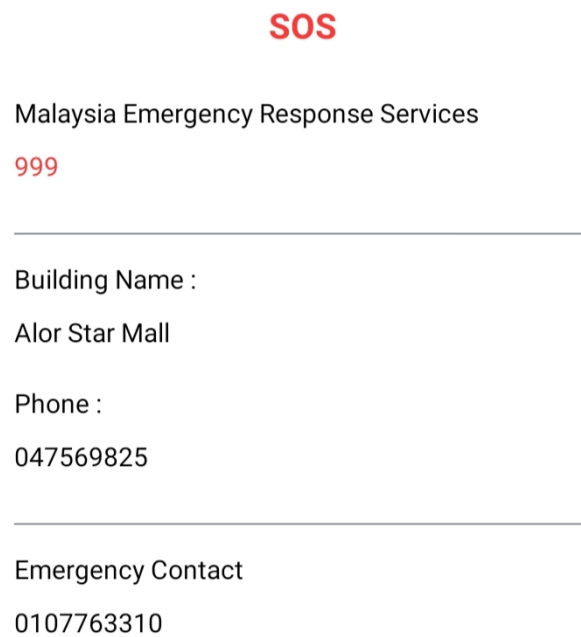
Password need to :
* be at least 8 characters.
* include upper case and lower case character.
* include at least 1 number.
* include at least 1 special character.

CONFIRM

Figure 4.12: User Profile Page

4.1.9 SOS

When user face an emergency problem, user can SOS card in the Home page and it will redirect user go to this page which will show in figure 4.13. User can click the phone number to make a phone call. The building information will only show when the user is already entered in the building. The emergency contact will only show when user is already add the emergency contact.



The image shows a mobile application interface for emergency services. At the top, the word "SOS" is displayed in large red letters. Below it, the text "Malaysia Emergency Response Services" is shown in black, followed by the red number "999". A horizontal line separates this header from the main content area. The main content area contains two sections. The first section is labeled "Building Name :" and shows "Alor Star Mall". The second section is labeled "Phone :" and shows "047569825". Another horizontal line separates this from the bottom section, which is labeled "Emergency Contact" and shows "0107763310".

SOS

Malaysia Emergency Response Services

999

Building Name :
Alor Star Mall

Phone :
047569825

Emergency Contact
0107763310

Figure 4.13: SOS Page

4.2 Testing Plan and Testing Output

Testing is the process of showing the correctness of the program. Need to be tested to show completeness, it can improve the quality of the software and provide maintenance assistance. Therefore, some test standards are needed to reduce test costs and operating time. The test software extends to the entire coding phase, and it represents the final review of configuration, design, and coding. Based on how the software reacts to these tests, we can decide whether to build a configuration in the study. All components of the application have been tested because failure to do so will result in a series of errors after using the software.

Table 4.1: Table of User Registration Testing Plan and Output

Input Value	Testing Plan	Output
Username, Email, Phone Number	Should be not null, not match with database	Warning message will show when it is null
Password, Confirm Password	Password and confirm password should match and password format must be correct	Warning message will show when it is null, wrong password format and password not match
Confirm Button	Successfully register and direct user back to Login Page	Warning message will show when fail to update or user is existed in the database

Table 4.2: Table of User Login Testing Plan and Output

Attribute	Testing Plan	Output
Username, Password	Should match with the database	Warning message will show when the username and password is not

		match with the database
Login Button	User exits and direct user to Home Page	Directed user to Home Page

Table 4.3: Table of User Detail Testing Plan and Output

Attribute	Testing Plan	Output
Username	Should match with the database and return user information	Display user information
New Password, New Confirm Password	New password and new confirm password should match and password format must be correct	Warning message will show when wrong password format and password not match
Confirm Button	Successfully update user information show direct back to Login page	Updated user information and directed user back to Login Page

Table 4.4: Table of ParkNow Testing Plan and Output

Attribute	Testing Plan	Output
Building Name, Username	Match with database and update user parking detail	If building name is not existed in database a warning message will show otherwise will update user parking detail

Table 4.5: Table of ExitNow Testing Plan and Output

Attribute	Testing Plan	Output
Building Name, Username, Payment Status	Match with database, payment status must be '0' and update user parking detail	If the payment status is '1' a warning message will show to user and user cannot scan building QR Code otherwise will update user parking detail

Table 4.6: Table of Building Parking Dashboard Testing Plan and Output

Attribute	Testing Plan	Output
Building Name	Match with database and return building's level id list	If building name not found will show warning message otherwise will return level id list
Building Name, Level ID	Match with database and return parking slot's status is not occupied list	If building name and level id is not found will show warning message otherwise will parking slot's status is not occupied list
Search Button	Must select Level ID and return parking slot's status is not occupied list	If not select level id a warning message will show

Table 4.7: Table of Building Report Parking Testing Plan and Output

Attribute	Testing Plan	Output
Card Plate Number, Parking Slot, Report Type	Should be not null, and parking slot should match with database	Warning message will show when it is null or the parking slot is not match with database
Confirm Button	Show success message and direct user back to Building Info Page	Success message will show and directed user back to Building Info Page

Table 4.8: Table of User Parking Detail Testing Plan and Output

Attribute	Testing Plan	Output
Scan Parking Slot QR	Parking slot should match with database and update user parking detail	Warning message will show when the parking slot is not existed in database otherwise will update user parking detail
Pay Button	After click must direct to Payment Method list Page	Directed user to Payment Method List Page
Select Payment Type	Should show success message	Success message will show when click the payment type

Table 4.9: Table of SOS Testing Plan and Output

Attribute	Testing Plan	Output
Phone Number	Click phone number must be call out	Will call out when click the phone number

4.3 Main Function Codes

4.3.1 User Registration

Figure 4.14 show coding before add a new user, will check the all the attribute is not null and password format is correct.

```
private String validateUserInformation() {
    String password = etPassword.getText().toString();

    if (etUsername.getText().toString().isEmpty()) {
        return "Username cannot be empty";
    } else if (etEmail.getText().toString().isEmpty()) {
        return "E-mail Address cannot be empty";
    } else if (etPhone.getText().toString().isEmpty()) {
        return "Phone number cannot be empty";
    } else if (password.isEmpty()) {
        return "Password cannot be empty";
    } else if (password.length() < 8) {
        return "Password length must longer than 8";
    } else if (password.length() > 15) {
        return "Maximum password length is 15";
    } else if (etConfirmPassword.getText().toString().isEmpty()) {
        return "Confirm Password cannot be empty";
    }

    boolean hasDigit = false;
    boolean hasAlphabet = false;
    boolean hasSymbol = false;

    for (int i = 0; i < password.length(); i++) {
        if (Character.isDigit(password.charAt(i))) {
            hasDigit = true;
        } else if (Character.isAlphabetic(password.charAt(i))) {
            hasAlphabet = true;
        } else if (String.valueOf(password.charAt(i)).matches("[^A-Za-z0-9]")) {
            hasSymbol = true;
        }
    }

    if (!hasDigit) {
        return "Password must contain at least 1 digit";
    } else if (!hasAlphabet) {
        return "Password must contain at least 1 alphabet";
    } else if (!hasSymbol) {
        return "Password must contain at least 1 special symbol";
    } else if (!password.equals(etConfirmPassword.getText().toString())) {
        return "Password not same";
    } else {
        return "";
    }
}
```

Figure 4.14: Check user attribute code

Figure 4.15 show coding, when all the attribute is validated then check the new user is already existed in database or not.

```
DocumentReference documentReference = firebaseFirestore.collection("User").document(etUsername.getText().toString());
documentReference.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DocumentSnapshot> task) {
        loadingDialog.dismiss();
        if (task.isSuccessful()) {
            DocumentSnapshot document = task.getResult();
            if (document != null) {
                if (document.exists()) {
                    defaultToast.warningToast("User had been existed");
                } else {
                    register();
                }
            } else {
                defaultToast.warningToast("User had been existed");
            }
        }
    }
});
```

Figure 4.15: Check user existed code

Figure 4.16 show coding, when new user is not existed then create the new user.

```
firebaseFirestore = FirebaseFirestore.getInstance();
User user = new User(
    etUsername.getText().toString(),
    etEmail.getText().toString(),
    etPhone.getText().toString(),
    etPassword.getText().toString(),
    "",
    "",
    "",
    "0",
    0
);

firebaseFirestore.collection("User").document(etUsername.getText().toString()).set(user)
    .addOnCompleteListener(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            loadingDialog.dismiss();
            if (task.isSuccessful()) {
                defaultToast.successToast("Congratulations");
                Intent intent = new Intent(RegisterActivity.this, LoginActivity.class);
                startActivity(intent);
                finish();
            } else {
                defaultToast.dangerToast("Fail to register. Please try again");
            }
        }
    });
}
```

Figure 4.16: Add new user

4.3.2 User Login

Figure 4.17 show coding, before user login will check the user is existed or not. If is existed then directed user to Home Page.

```
firebaseFirestore.collection("User")
    .whereEqualTo("Username", etUsername.getText().toString())
    .get()
    .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            enableButtonLogin();
            loadingDialog.dismiss();

            if (task.getResult() == null) {
                defaultToast.dangerToast("User not found");
                return;
            }

            if (task.isSuccessful()) {
                for (QueryDocumentSnapshot document : Objects.requireNonNull(task.getResult())) {
                    if (!etPassword.getText().toString().equals(String.valueOf(document.getData().get("password")))) {
                        defaultToast.warningToast("Username or password wrong");
                        return;
                    }

                    defaultToast.successToast("Successful Login");
                    login(document);
                }
            } else {
                defaultToast.dangerToast("User not found");
            }
        }
    });
```

Figure 4.17: Check user existed code

4.3.3 ParkNow

Figure 4.18 show coding, before update user parking detail will check the building name is existed or not.

```
firebaseFirestore.collection("Building")
    .whereEqualTo("building_name", buildingName)
    .get()
    .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            loadingDialog.dismiss();

            if (Objects.requireNonNull(task.getResult()).isEmpty()) {
                defaultToast.warningToast("Building not found");
                return;
            }

            if (task.isSuccessful()) {
                for (QueryDocumentSnapshot document : Objects.requireNonNull(task.getResult())) {
                    setBuildingInfo(document);
                }
            } else {
                defaultToast.dangerToast("User not found");
            }
        }
    });
```

Figure 4.18: Check building existed code

Figure 4.19 show coding, if the building name is existed then update the user parking detail.

```
DocumentReference documentReference = firebaseFirestore.collection("User").document(username);
documentReference
    .update(
        "building_name", buildingName,
        "enter_at", String.valueOf(System.currentTimeMillis() / 1000L),
        "payment_status", 1)
    .addOnCompleteListener(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            loadingDialog.dismiss();

            if (task.isSuccessful()) {
                SharedPreferences.Editor edit = sharedPreferences.edit();
                edit.putInt("payment_status", 1);
                edit.apply();

                defaultToast.successToast("Welcome To " + buildingName);
                Intent intent = new Intent(ScanBuildingQRCodeActivity.this, HomeActivity.class);
                startActivity(intent);
                finish();
            } else {
                defaultToast.successToast("Fail to update");
            }
        }
    });
```

Figure 4.19: Update user parking detail code

4.3.4 ExitNow

Figure 4.20 show coding, before update user parking detail will check the building name is existed or not.

```
firebaseFirestore.collection("Building")
    .whereEqualTo("building_name", buildingName)
    .get()
    .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            loadingDialog.dismiss();

            if (Objects.requireNonNull(task.getResult()).isEmpty()) {
                defaultToast.warningToast("Building not found");
                return;
            }

            if (task.isSuccessful()) {
                for (QueryDocumentSnapshot document : Objects.requireNonNull(task.getResult())) {
                    setBuildingInfo(document);
                }
            } else {
                defaultToast.dangerToast("User not found");
            }
        }
    });
```

Figure 4.20: Check building existed code

Figure 4.22 show coding, if the building name is existed then update the user parking detail.

```
DocumentReference documentReference = firebaseFirestore.collection("User").document(username);
documentReference
    .update(
        "building_name", "",
        "parking_slot_id", "",
        "enter_at", "0")
    .addOnCompleteListener(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            loadingDialog.dismiss();

            if (task.isSuccessful()) {
                SharedPreferences.Editor edit = sharedPreferences.edit();
                edit.remove("building_name");
                edit.putInt("payment_status", 0);
                edit.apply();

                defaultToast.successToast("Thank You");
                Intent intent = new Intent(ScanBuildingQRCodeActivity.this, HomeActivity.class);
                startActivity(intent);
                finish();
            } else {
                defaultToast.warningToast("Fail to update");
            }
        }
    });
};
```

Figure 4.21: Update user parking detail

4.3.5 Building Parking Dashboard

Figure 4.23 show coding, once user go in this Parking Dashboard will use building name get all the building's level id list.

```
DocumentReference documentReference = firebaseFirestore.collection("Building").document(buildingName);
documentReference.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DocumentSnapshot> task) {
        loadingDialog.dismiss();
        levelList.add("Select");
        if (task.isSuccessful()) {
            DocumentSnapshot documentSnapshot = task.getResult();
            assert documentSnapshot != null;
            if (documentSnapshot.exists()) {
                try {
                    JSONArray levelJA = new JSONArray(String.valueOf(Objects.requireNonNull(documentSnapshot.getData()).get("parking_level")));
                    for (int i = 0; i < levelJA.length(); i++) {
                        levelList.add(String.valueOf(levelJA.get(i)));
                    }

                    setSpinner(levelList);
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            } else {
                defaultToast.warningToast("No Level Found");
            }
        } else {
            defaultToast.warningToast("No Level Found");
        }
    }
});
};
```

Figure 4.22: Get building level list code

Figure 4.24 show coding, user select level id it will trigger this function to get all the parking slot that under this level id.

```
firebaseFirestore.collection("Building").document(buildingName).collection(levelId)
    .whereEqualTo("slot_status", 0)
    .get()
    .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            loadingDialog.dismiss();

            if (Objects.requireNonNull(task.getResult()).isEmpty()) {
                defaultToast.dangerToast(levelId + " parking slot not found");
                return;
            }

            if (task.isSuccessful()) {
                for (QueryDocumentSnapshot document : task.getResult()) {
                    parkingSlotList.add(document.toObject(ParkingSlot.class));
                }

                if (parkingSlotList.size() != 0) {
                    recyclerView.setVisibility(View.VISIBLE);
                    layoutNoInfo.setVisibility(View.GONE);
                } else {
                    recyclerView.setVisibility(View.VISIBLE);
                    layoutNoInfo.setVisibility(View.VISIBLE);
                    tvInfo.setText(R.string.no_parking_slot_found_content);
                }

                parkingDashboardAdapter.notifyDataSetChanged();
            } else {
                defaultToast.dangerToast(levelId + " parking slot not found");
            }
        }
    });
```

Figure 4.23: Get parking slot according to selected level

4.3.6 Building Report Parking

Figure 4.26 show coding, user click confirm button will validate all the attribute.

```
btnConfirm.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        String selectedReportType = spinner.getSelectedItem().toString();  
        if (etCarPlateNumber.getText().toString().equals("")) {  
            defaultToast.warningToast("Car plate number cannot be empty");  
            return;  
        } else if (etParkingSlot.getText().toString().equals("")) {  
            defaultToast.warningToast("Parking slot cannot be empty");  
            return;  
        } else if (selectedReportType.equals("Select")) {  
            defaultToast.warningToast("Please select 1[One] report type");  
            return;  
        }  
  
        checkParkingSlot(etParkingSlot.getText().toString());  
    }  
});
```

Figure 4.24: Validate all attribute code

Figure 4.27 show coding, after validation will check the parking slot is existed or not.

```
DocumentReference documentReference = firebaseFirestore.collection("Building").document(buildingName).collection(levelId).document(parkingSlotId);  
documentReference.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {  
    @Override  
    public void onComplete(@NonNull Task<DocumentSnapshot> task) {  
        loadingDialog.dismiss();  
  
        if (task.isSuccessful()) {  
            DocumentSnapshot document = task.getResult();  
            if (document != null) {  
                if (document.exists()) {  
                    defaultToast.successToast("Thank you. We will take action as soon as possible.");  
  
                    Intent intent = new Intent(BuildingReportActivity.this, BuildingActivity.class);  
                    startActivity(intent);  
                    finish();  
                } else {  
                    etParkingSlot.setText("");  
                    defaultToast.warningToast("Parking slot not exists");  
                }  
            } else {  
                etParkingSlot.setText("");  
                defaultToast.warningToast("Parking slot not exists");  
            }  
        } else {  
            etParkingSlot.setText("");  
            defaultToast.dangerToast("Building not found");  
        }  
    }  
});
```

Figure 4.25: Validate parking slot code

4.3.7 User Parking Detail

Figure 4.28 show coding, once user go in this User Parking Detail Page will use username get user parking detail and when it is success get the data figure 4.29 show will calculate the total parking fee amount that user need pay.

```
firebaseFirestore.collection("User")
    .whereEqualTo("username", username)
    .get()
    .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            loadingDialog.dismiss();

            if (Objects.requireNonNull(task.getResult()).isEmpty()) {
                defaultToast.dangerToast("User not found");
                return;
            }

            if (task.isSuccessful()) {
                for (QueryDocumentSnapshot document : Objects.requireNonNull(task.getResult())) {
                    tvBuildingName.setText(String.valueOf(document.getData().get("building_name")));
                    tvParkingSlot.setText(String.valueOf(document.getData().get("parking_slot_id")));

                    long milliSeconds = Long.parseLong(String.valueOf(document.getData().get("enter_at"))) ;
                    if (milliSeconds != 0) {
                        SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.ENGLISH);
                        Date resultDate = new Date(milliSeconds * 1000L);
                        tvEnterTime.setText(simpleDateFormat.format(resultDate));

                        if (!tvParkingSlot.getText().toString().equals("")) {
                            ivScan.setVisibility(View.GONE);
                        }

                        calculateTotalAmount(milliSeconds, chargePerHours);
                    }
                }
            } else {
                defaultToast.dangerToast("User not found");
            }
        }
    });
```

Figure 4.26: Get user parking detail code

```
long currentMilliSeconds = System.currentTimeMillis() / 1000L;
long totalHoursPark = (currentMilliSeconds - milliSeconds) / (1000 * 60);
long totalAmount;

if (totalHoursPark == 0) {
    totalAmount = chargePerHours;
} else {
    totalAmount = ((currentMilliSeconds - milliSeconds) / (1000 * 60)) * chargePerHours;
}

tvTotalAmount.setText(String.valueOf(totalAmount));
```

Figure 4.27: Calculate total parking fee amount

4.3.8 SOS

Figure 4.30 show coding, when the user clicks the phone number it will direct call out the phone number.

```
if (Build.VERSION.SDK_INT > 22) {  
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.CALL_PHONE) != PackageManager.PERMISSION_GRANTED) {  
        ActivityCompat.requestPermissions(SOSAlertActivity.this, new String[]{Manifest.permission.CALL_PHONE}, 101);  
        return;  
    }  
  
    Intent callIntent = new Intent(Intent.ACTION_CALL);  
    callIntent.setData(Uri.parse("tel:" + phoneNumber));  
    startActivity(callIntent);  
} else {  
    Intent callIntent = new Intent(Intent.ACTION_CALL);  
    callIntent.setData(Uri.parse("tel:" + phoneNumber));  
    startActivity(callIntent);  
}
```

Figure 4.28: Phone call code

CHAPTER 5

SUMMARY AND CONCLUSION

5.1 Summary

The increasing of car usage is growing rapidly, and also the system of the parking lots must upgrade to be more systematic and reliable to the users. The Parking Management Application is developed to perform the effectiveness of the system and minimize the time and energy of users in order to find the parking space. The application can identify the total quantity of available parking slots and direct make payment in this application.

5.2 Limitations of The System

There are several limitations that occurred throughout the development of the project in order to achieve the objectives. These limitations of this system are can be used only for car. It is not suitable for others vehicles. Besides, this system only can be applied for building only and it is not suitable for street parking. This is because, this system needs a network connection in Ultrasonic Sensor to check the availability of parking space.

5.3 Future Development

5.3.1 Empty slot booking

Users able to book the empty parking slot before they go to building. Users can view and select the empty parking slot that they preferred and book the parking slot.

5.3.2 Location reminder

Through the Parking Management application, users able to get a location reminder on which area their parked and even on which exact parking slot they parked their cars.

REFERENCES

Lyaka Beni (n.d). Smart Parking System. Retrieved from:

https://www.academia.edu/34838309/Smart_Parking_System.pdf

Aekarat Seliw, Watcharasuda Hualkasin, Supattra Puttinaovarat, Kanit Khaimook(2019), Smart Car Parking Mobile Application base on RFID and IoT. Retrieved from:

https://www.researchgate.net/publication/333247166_Smart_Car_Parking_Mobile_Application_base_d_on_RFID_and_IoT

Ms. Marzia Alam(2017), Automated Car Parking System. Retrieved from:

http://dspace.bracu.ac.bd/xmlui/bitstream/handle/10361/8412/11121061%2C10121040_EEE.pdf?sequence=1&isAllowed=y

N. Adburrahman Izzuddin B Nik Ismail (2018), Smart Entry Parking System. Retrieved from:

<https://myfik.unisza.edu.my/www/fyp/fyp17sem2/report/041124.pdf>

Ndayisaba Corneille (2018), Online Vehicle Parking Reservation System. Retrieved from:

https://www.researchgate.net/publication/324517934_ONLINE_VEHICLE_PARKING_RESERVATION_SYSTEM#pff

Free Projectz (n.d), Project Report On Online Car Parking System, Retrieved from:

<https://www.freeprojectz.com/project-report/3030>

Geeta Kamaraj (2017), Parking Management System, Retrieved from:

https://www.academia.edu/36410792/Parking_Management_System_Parking_Management_System

Ray Adderley, S. S. M. Fauzi, M. N. F. Jamaluddin, M. A. B. Marizalee(2019), Design and Development of Mobile Application for Parking Reservation System. Retrieved from:

https://www.researchgate.net/publication/337244078_Design_and_Development_of_Mobile_Application_for_Parking_Reservation_System

S. Gatalwar, R. Agnihotri, N. Gujarathi, A. Behere (2016), ParkSmart: Android Application for Parking System. Retrieved from: <http://ijcsn.org/IJCSN-2016/5-1/ParkSmart-Android-Application-for-Parking-System.pdf>

M. Na'imullah Bin Jamaludin(2013), Mobi Parking Navigator System. Retrieved from:

<http://umpir.ump.edu.my/id/eprint/9460/1/CD8306.pdf>