PARKING RESERVATION AND MANAGEMENT SYSTEM FOR UTAR KAMPAR CAMPUS

BY

KUEH WEE XIN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)
Faculty of Information and Communication Technology
(Kampar Campus)

JUNE 2025

PARKING RESERVATION AND MANAGEMENT SYSTEM

FOR UTAR KAMPAR CAMPUS

BY

KUEH WEE XIN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)
Faculty of Information and Communication Technology
(Kampar Campus)

JUNE 2025

COPYRIGHT STATEMENT

This Final Year Project report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor, Prof. Ts. Liew Soung Yue for his continuous support throughout the whole project. His professional guidance and the constructive feedback helped me a lot whenever I encountered challenges during this project. I would also want to express my sincere thanks to my parents, family and friends for their support and encouragement throughout this journey.

ABSTRACT

This project is dedicated to developing a smart parking system at UTAR Kampar campus. The difficulty in finding an optimal parking space has always been a problem for campus users who drive. The increasing demand for parking spaces has resulted in a waste of time and traffic congestion, especially during peak hours. Therefore, research is conducted in this report to examine existing parking systems, parking space availability detection methods, recommendation system, and reservation strategies. Based on the literature review, the proposed solution is to develop a mobile-based parking reservation and management system to address traffic congestion, reduce search time and solve parking inefficiency. The system will give recommendations to users, enable them to reserve parking using credit, view real-time availability of spaces, and receive GPS navigation to their reserved spots. The proposed system is developed using Flutter for the user interface with integration of Firebase and the Google Maps service into a mobile application to demonstrate how smart reservations can reduce search times, optimize parking resources in campus and improve user experience.

Area of Study: Mobile Application Development

Keywords: Mobile Application, Smart Parking, Reservation System, Recommendation Algorithm, Credit Management

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

IoT	Internet of Things
IPS	Intelligent Parking System
AVL	Adelson-Velsky and Landis
YOLO	You Only Look Once
IPS	Indoor Positioning System
CNNS	Convolutional Neural Networks
HOAs	Residential Community
IR	Infrared

CHAPTER 1

Introduction

Parking management has become one of the most pressing problems in urban life and institutional settings. Hence, it is important to ensure the efficient use of limited parking resources to alleviate traffic congestion caused by people encroaching upon certain areas trying to look for a place to park [1]. At UTAR Kampar campus, the increasing demand for parking has voiced the need for better development. Traffic congestion often occurs during peak hours because many people seek to park in the same parking zone with limited spaces. They must go to the area on-site in order to look for the available empty spaces.

To overcome this problem, the project proposes an efficient parking reservation and credit system that enables users to make reservations and manage their parking credits efficiently. This system aims at making optimal use of the available sparse parking resources by allowing the reservation of the parking places from a single access point. Users will also be able to utilize parking credits which will act as a means for making reservations on arrival and making sure an appropriate parking spot is designated. The credit management system enforces fair usage and discourages unnecessary reservations to ensure better slot availability across the campus.

The system integrates real-time parking availability indications, reservation, destination navigation, and credit management. All these features form a comprehensive smart parking solution that helps in improving parking efficiency and management in campus.

1.1 Problem Statement

At UTAR Kampar campus, students and staff always struggle with finding available parking slots during peak hours as the demand is highest. The layout of the campus exacerbates the issue because the parking areas offer limited spaces and are located at varying distances from lecture halls and other campus infrastructure. This creates a dilemma: students either spend valuable time searching for a parking spot in the parking zone that is near their destinations or are forced to park in more distant areas with better availability. Hence, recurring frustration and stress are a frequent occurrence that wastes time but also in some instances to make students late for classes. Now, there is no system or application within campus that could let a user check the parking slots occupancy in real time or even let the user reserve a parking slot. In this manner, the present parking management is inefficient; it just adds more inconveniences and daily frustration to students and staff, pointing each time toward the idea of recurrence of problems with solutions that are more organized, and technology based.

1.2 Objectives

i. Alleviate traffic congestion

The primary goal is to reduce traffic congestion related to parking. This objective centers on minimizing the time drivers spend searching for available parking spaces, eliminating the need for vehicles to circle through parking areas repeatedly. The purpose also includes enabling drivers to secure parking spaces efficiently, reducing uncertainty and search time. Ultimately, the aim is to decrease the number of vehicles wandering through parking areas, thereby reducing overall traffic congestion in and around parking facilities.

ii. Improve parking resource management through a credit-based system

The second goal is to improve parking resource management through a credit-based system. This objective aims to enable fair and controlled access to parking spaces by allowing users to utilize allocated credits for reservations. The purpose includes establishing accountability for reservation usage to ensure equitable allocation of parking resources among all users. Ultimately, this goal seeks to achieve efficient utilization of available parking resources, preventing waste and maximizing the benefit of existing parking infrastructure for the entire user community.

iii. Enhance user experience through smart navigation and recommendations

The third goal is to enhance user experience through smart navigation and recommendations. This objective aims to provide seamless guidance for users to reach their designated parking locations efficiently. The purpose includes offering intelligent parking suggestions that reduce unnecessary driving and minimize fuel consumption by eliminating the time users would otherwise spend wandering to locate their reserved spaces. Ultimately, this goal seeks to create a more convenient and environmentally conscious parking experience that saves both time and resources for users.

1.3 Project Scope

The project focus is developing a mobile-based parking reservation and management system for the UTAR Kampar campus. The final deliverable will be a functional prototype built using Firebase, Flutter, and Google Maps API. The system is designed for both Android and iOS platforms, and its core features include real-time parking space availability display, parking slot recommendations, slot reservations, GPS navigation to the reserved destination and the integration of credit system.

Users can inspect the availability of parking spaces in different zones through the mobile application and reserve empty slots using credit instead of traditional payment methods. Rules and penalties will be implemented to discourage misuse, such as no-shows or late cancellations. The system will integrate Google Maps API to provide turn-by-turn navigation guiding users to their reserved parking space. This feature aims to reduce the searching time for empty parking spaces and help reduce traffic congestion on campus.

Furthermore, an intelligent recommendation system will also be featured in this project to give recommendation in the optimal parking slots. This helps user in decision making by recommending the most convenient parking options and enhancing the overall user experience. When they arrive at the destination, users would need to check in at the reserved slot through QR code verification. The system will determine real-time slot based on the reservation record in place of actual IoT sensor integration.

1.4 Motivation

The inefficiency in parking management and the limited parking space at UTAR Kampar campus have created significant inconvenience for students and staff who drive during peak hours. Due to the high demand for parking, campus users are forced to waste valuable time searching for available slots which make them frustrated. A reservation-based system may help solve the problems by providing users with prior knowledge on their car parks, thereby leading towards easier campus life. This project aims to enhance the efficiency of parking operations through a reservation system that allows campus users to secure their parking space. Users would be able to utilize allocated credit points for reserving parking places thus encouraging equity utilization of scare parking resources. Further another aspect includes this system's rules and penalties purposefully established so as discourage misconduct like reserving without using or even cancelling too late reservation. In this way it ensures that parking is controlled in an effective manner.

1.5 Contributions

This project contributes a different approach to parking management by implementing a reservation and management system customized for UTAR Kampar campus. The target user of this system are all the campus users who drive. Unlike other existing parking systems that rely on real-time availability without reservations, this system introduces slot reservation features that allow users to secure parking spaces in advance and promote the efficient use of parking resources. A key project contribution is the implementation of smart parking through the integration of technologies to develop a mobile app that supports various features aimed at enhancing parking efficiency and operational management on campus. A notable part of the system is the intelligent slot recommendation algorithm which recommends the optimal parking space to user. The system also allows users to inspect the slot availability in real-time and reduce the search time for parking. Rules and penalties in the system aimed at curbing misuse, for users who fail to use their reserved spots or cancel reservations too late may face penalties to ensure all the spaces are used efficiently. This rule for fair use in this system prevents users from reserving spots unnecessarily and promotes the efficient use of parking resources. By introducing a smart parking solution to UTAR Kampar campus, this system not only improves parking experience for all campus users but also provides a framework that can be adapted for other institutions which deal with same parking problems.

1.6 Report Organization

There are total of seven chapters in each report, each building upon the previous chapter to demonstrate the development process step by step. Chapter 1 introduces the project by outlining its background, problem statement, objectives, scope, motivation, and contributions. Chapter 2 presents the literature review, which covers research on reservation systems, recommendation algorithms, real-time parking availability detection methods, and existing systems. This chapter helps to identify relevant technologies and approaches to be incorporated into the proposed solution. Chapter 3 outlines the proposed system solution based on the findings from Chapter 2. It includes the system requirements, use case diagrams and descriptions, the system architecture diagram, and the project timeline. The fourth chapter demonstrates the system design, covering the system flow diagram, core component specifications, and database structure. Implementation details are covered in Chapter 5, it explains the system implementation, starting from the software setup, configuration, and tools used, to data collection methods and system operation, including user interface design and backend logic implementation. Chapter 6 presents comprehensive testing, including both system and application testing, to ensure the system functions as expected. The final chapter concludes with the report by summarizing the project and providing recommendations for future improvements.

CHAPTER 2

Literature Review

This literature review seeks to analyze a few important themes that are critical to the development of a parking reservation and management system. The review has four main sections: recommendation systems, reservation systems, real-time parking availability detection and review of existing systems. Each theme has been carefully analyzed with a view of providing an extensive understanding of the current state of research and technology in these areas. By investigating these aspects, this review aims to understand how existing systems and technologies can be integrated and adapted to improve parking management on campus.

2.1 Recommendation system

Research on parking recommendation systems sought to enhance parking efficiency and resource utilization. One such system focuses on the decision-making behaviors and psychological characteristics of travelers to give intelligent parking recommendations as developed in [2]. This was seen as an intelligent parking service (IPS) which guided users to appropriate spaces in urban centers by considering price, distance and driving time. The integration of real-time data collection with cloud computing adds strength to this system allowing for informed decisions on where to park based on current availability [3].

Among its strengths are the inclusion of real-time data and consideration of factors influencing the user's parking choice such as cost or closeness to destination. With this, time wasted searching for car parks has greatly reduced, leading to lesser traffic jams while improving user satisfaction.

Nevertheless, despite these advancements, an important limitation is the absence of psychological aspects in several systems. For instance, although user's preferences such as cost and distance are considered, other barriers related to mentality like distress tolerance or anxiety while parking is not workable. To fill this gap [2] presents a model for recommendations that takes into consideration the psychology behind them by integrating individuals' psychological thresholds and recommending models that change with time based on the usage of parking lots

by different drivers. In this way, customization becomes more confined due to adaptation with user behavior.

Moreover, Canli et al. offer an AVL tree-based method that enables the efficient allocation and reservation of parking lots. By virtue of their self-balancing property, AVL trees permit the system to swiftly search through adjacent parking areas as well as locate the most appropriate slots that are vacant. In addition, the process is enhanced through integration with IoT sensors which provide real-time information on free spaces for cars to fit in and stay there. Among other things this method has more efficient searching capabilities as well as saves energy with an improvement rate of 99% over standard hierarchical or non-hierarchical models from various simulations [4].

However, the AVL tree-based system still has a major drawback: the problem of maintaining balanced binary trees [4]. This can be slow, especially for real-time applications. It is a big problem when it comes to scalability and responsiveness of systems. The solution would lie in finding other data structures or optimization techniques that will reduce computation time without affecting accuracy or efficiency.

While the current parking recommendation systems have made a lot of efforts in enhancing user experience by using real time data and intelligent recommendations. But some areas are still weak. The important steps toward creating more effective, user centric systems would be to broaden psychological factors and handle computational problems arising from advanced data structures like AVL trees. If well executed these improvements could completely change the way parking recommendation technologies perform in future and increase their scalability.

2.2 Reservation

Although recommendation systems are important in helping users locate the best parking slots, the whole efficiency of the system is highly reliant on how effective a reservation mechanism is. According to [5], parking reservation techniques are crucial for city parking resource management. They usually utilize real-time processing via mobile applications and sensor networks that enable users to check for availability of parking lots and subsequently book space accordingly. This capability enhances not only the overall efficiency of parking management but also leads to a better allocation of resources.

A major advantage pointed out by [5] is that inventory control methods may be used to allocate parking spaces. These methods involve grouping parking lots according to vehicle categories such as compact cars or electric ones making sure each category has its designated area. This method of allocation considers multiple factors, including the timing and number of parking requests, characteristics of parking lots, and driver preferences. Furthermore, an integration approach incorporating dynamic pricing models into demand management for optimizing the use of parking spaces is embraced herein. The pricing system adjusts rates in relation to demand encouraging optimal utilization of such resources, especially during busier hours thus enhancing urban parking management overall [5].

Despite these strengths, many problems still exist. One major issue is the increasing demand for parking spaces and the need to ensure fair allocation across different vehicle types, which can result in longer search times and inefficient use of resources. For example, if compact cars or electric vehicles were prioritized based on their availability then it might put at stake other types of cars leading to uneven parking proportions. There are also inequities that dynamic pricing may give rise to in so far as demand management is concerned because some users may be priced out of more convenient parking places thus lowering their overall satisfaction.

In order to tackle such issues further research can be conducted into the methods that would enhance justice in allocation of bays whilst ensuring that the system remains functional. This could involve designing more sophisticated algorithms that would alter allocations dynamically depending on real time needs per type of vehicle. Another possible remedy would be to incorporate predictive analytics into the system such that demand for parking spaces in each

area could be estimated and spaces allocated accordingly thus minimizing search time and maximizing usage efficiency. Besides, feedback from users can contribute to a better distribution process since their different requests regarding parking are considered.

Although reservation systems offer considerable advantages in optimizing parking space utilization and managing demand through real-time data and dynamic pricing, their effectiveness can be limited by issues related to fair allocation and rising demand. Addressing these challenges requires a more adaptive approach that considers a broader range of factors, such as vehicle types, user preferences, and demand forecasting. Enhancing these systems will improve the user experience and further optimize the use of available parking resources in urban environments [5].

2.3 Available Parking Space Detection

Accurate detection of available parking spaces is a critical requirement for both parking reservation systems and recommendation systems to function effectively. In this project, different types of detection methods is reviewed such as Internet of Things (IoT) for sensor-based detection and machine learning algorithms like YOLO (You Only Look Once) for camera-based detection.

2.3.1 IoT-based Systems

The Indoor Positioning System (IPS) framework developed by the researchers in [3] relies upon IoT components such as Raspberry Pi, NodeMCU, RFID and infrared (IR) sensors. This system collects and processes real time data on the slot occupancy and informs users on their mobile applications about the nearest available ones. Consequently, the search time is significantly decreased, and parking violations prevented. Similarly, Fahim et al utilized various sensors including IR, ultrasonic sensors and camera-based systems to capture occupancy of a parking spot in real-time. It is through these wireless networks that these sensors are interconnected allowing them to communicate perfectly with central management systems [6].

One other instance of achieving similar goals using IoT is shown in [7] where ultrasonic and infrared sensors are used for vehicle detection in a parking area. For instance, ultrasonic sensors measure distances between vehicles so that when no vehicle is detected; then it implies that there must be an unoccupied space while infrared responds appropriately thus making vehicle detection more reliable due to its fewer effects from environmental factors. Moreover, RFID technology is also incorporated since RFID readers will track cars as they come into or leave the lot by scanning RFID tags attached to them via one of the latter methods mentioned above. Microcontrollers, such as Arduino, ESP32, and Raspberry Pi, interface with these sensors to process data locally, offering additional computational power for tasks like running local servers or managing databases. Wi-Fi enables data transmission from the microcontroller to cloud services like Firebase, ensuring real-time updates.

Despite the promising results, the study highlights several limitations. The accuracy of data collected by IoT sensors may be compromised by environmental conditions and hardware malfunctions. Additionally, reliance on real-time cloud databases raises concerns about data

security and privacy. To mitigate these limitations, the researchers suggest integrating the advanced machine learning algorithms for higher data accuracy and better predictive capabilities.

2.3.2 YOLO-based Detection

Computer vision-based approaches offer an alternative to sensor-based parking detection systems. According to [8], a modified YOLO architecture called T-YOLO has been developed specifically for detecting vehicles in parking lots from overhead camera perspectives. This method addresses the challenge of monitoring large parking areas using cameras positioned at high distances. The entire parking spaces can be monitored with a single device. The T-YOLO model incorporates several key modifications to the standard YOLO-v5 architecture. The original Focus layer is replaced with a multi-scale module (MSM) that processes input images at different scales using bilinear interpolation and ENet initial blocks. This multi-scale approach enhances the detection of small and tiny vehicles that appear in overhead camera views. Additionally, the model integrates spatial-channel attention modules (SCAM) that help the network focus on relevant spatial and channel information, improving feature extraction for vehicle detection tasks.

Experimental results on the dataset show significant performance improvements, with T-YOLO achieving 96.34% precision compared to 63.87% for the YOLO-v5 model. The modified architecture maintains computational efficiency with 7.26 million parameters and operates at approximately 30 fps [8]. The model shows strength in detecting tiny vehicles from camera views, which is essential for practical parking monitoring systems. However, camera-based detection systems face their own limitations. The system's performance depends on environmental factors such as lighting variations, weather, and occlusions. Additionally, the system requires clear sight lines and may struggle with partially obscured vehicles or complex parking layouts.

2.3.3 Advantages and Disadvantages

There are many benefits of using IoT based vehicle detection like improved accuracy of real time monitoring, reduced parking search time and alleviation of traffic congestion. Additionally, these systems can integrate with smartphones to provide users with real time updates and navigational assistance through their mobile phones which enhances user experience. Nonetheless, bothersome issues regarding deployment costs and recurrent maintenance needs may make the IoT solution difficult to implement due to sensor inaccuracies resulting from weather extremes among others. Hence, there is a need for robust system design and careful planning for IoT-based vehicle detection to remain reliable and cost-effective in the long run.

The YOLO algorithm has specific advantages concerning faulty parking space detection via T-YOLO Tiny amongst them being ability process information almost instantly as well as an efficient output for devices working at edges with limited processing capacity. The model demonstrates high precision in detecting vehicles from overhead camera views and can operate effectively under various lighting conditions. However, camera-based systems face limitations including dependency on clear sight lines, potential performance degradation in adverse weather conditions, and challenges with complex parking layouts. Additionally, initial setup costs for camera infrastructure and the need for adequate lighting can present implementation barriers.

2.4 Review of existing system

2.4.1 Parkalot

Having reviewed various detection and reservation methods, examining existing parking management systems to identify their features and limitations is essential. One example is Parkalot, a web-based parking reservation system designed to address parking challenges in workplaces, residential communities (HOAs), and campuses. Parkalot simplifies parking management and allocation by providing a cloud-based solution that eliminates the need for physical hardware. Parking administrators can allocate spaces, define regulations, manage reservations, and report parking violations. Moreover, the system allows users to assign priority parking slots to individuals who need them the most [9].

One key advantage of Parkalot is its accessibility. Users are not required to install any software or applications, as it is web-based. This streamlined user experience enhances ease of use and reduces friction in accessing the service. Additionally, because the system is cloud-based and does not require any hardware, it is a cost-effective solution for organizations that do not want to invest in expensive infrastructure.

However, Parkalot has certain limitations. The system assumes that all parking spaces are available unless reserved. This approach requires users to actively reserve a spot in advance, which may not suit dynamic parking environments were parking availability changes rapidly. Furthermore, the system does not have a mechanism to confirm whether the reserved spot is occupied by the correct vehicle. It relies on users to manually report parking violations, which can lead to inefficiencies and misuse of parking spaces.

2.4.2 Flexi Parking

Another existing parking reservation system is Flexi Parking, a mobile-based parking management application widely used in Malaysia. It is designed to simplify on-street and municipal parking payments by digitizing transactions and removing the need for physical parking coupons or meters. Users can manage their parking sessions directly through their smartphones. They can register multiple vehicles, purchase credits, select parking duration, receive notifications before their session expires, and extend their parking remotely [10].

The main advantages of Flexi Parking include its convenience since users no longer need coins or paper tickets. The app also has broad coverage across multiple municipal councils in Malaysia. It helps users manage their time effectively through reminders and extensions. By reducing paper usage, it also lowers environmental impact and improves record-keeping for both users and authorities, which makes enforcement more efficient.

However, the system has limitations. It does not provide real-time data on parking space availability, so users may still struggle to find an empty spot even after paying. The app depends on internet connectivity and smartphone availability, which can limit usability in areas with poor network coverage or for users unfamiliar with mobile applications. Some users have also reported occasional technical issues such as difficulties with credit top-ups or vehicle registration, which can affect the overall experience.

2.4.3 JomParking

JomParking is a mobile application that focuses on streamlining parking payment processes through digital solutions. The system eliminates the need for traditional payment methods such as parking coupons and reduces waiting times at parking meters by enabling smartphone-based payments. JomParking provides real-time updates and instant notifications to keep users informed about their parking status throughout their parking session [11].

A key strength of JomParking is its practical approach to addressing immediate pain points in urban parking. The reminder feature alerts users 15 minutes before parking expiration, allowing them to extend their parking time remotely without returning to their vehicles. This functionality significantly enhances user convenience and reduces the risk of parking violations. The secure digital payment system also promotes cashless transactions, aligning with modern payment preferences while improving transaction efficiency.

Despite these advantages, JomParking's limitations include its focus primarily on payment processing rather than comprehensive parking management. The system does not address fundamental issues such as parking space availability or helping users locate empty spots, which are critical components of holistic parking solutions. Additionally, the system appears to be designed primarily for street parking scenarios rather than structured parking facilities like campus environments, potentially limiting its applicability to institutional settings where space allocation and reservation systems are more crucial.

2.4.4 Existing Parking Application Comparison

Table 2.4.4.1: Comparison table

Feature	Parkalot	Flexi Parking	JomParking
Platform	web-based	mobile application	mobile application
Target Environment	Workplaces, residential communities, campuses	On-street and municipal parking	On-street, off-street, private car parks, multiple cities
Reservation Capability	Yes	No	No (Immediate parking)
Real-time Parking Info	No	No	Limited
Navigation	No	No	Limited
Payment Method	Digital payment	Preloaded credit	Preloaded credit
User Convenience / Features	Assign priority slots, cloud-based management	Session reminders, remote extension, multi-vehicle support	Session reminders, remote extension, notifications, digital receipts

The comparison table summarizes the key features, advantages, and limitations of three existing parking management systems which are Parkalot, Flexi Parking, and JomParking. The table highlights differences in platform, target environment, reservation capability, real-time parking information, navigation, payment methods and user convenience.

Parkalot is a web-based system designed primarily for workplaces, residential communities, and campuses. It enables users to book specific parking spaces ahead of time. It is suitable for environments where allocation and prioritization of parking spaces are important. However, it lacks real-time occupancy information and navigation. Besides that, it relies on manual reporting for enforcement, which may limit its effectiveness in dynamic parking environments.

Flexi Parking is a mobile application focused on on-street and municipal parking. While it does not provide the ability to reserve a specific parking bay in advance, it offers session reminders, remote extensions, and multi-vehicle management, which enhance user convenience. Its reliance on preloaded credits and smartphone connectivity makes it suitable for cashless

parking but limits usability in areas with poor network coverage or for users unfamiliar with mobile apps.

JomParking is also a mobile application that allows users to initiate parking sessions immediately at a chosen location through the “Park Now” feature. It provides limited navigation and real-time information about parking areas, along with reminders and digital receipts. While the app improves the efficiency of parking payments, it is primarily focused on payment processing and does not support advanced features such as specific slot reservations or comprehensive occupancy tracking. Its suitability is therefore more aligned with street parking rather than structured environments like campuses.

Overall, the comparison highlights that each system addresses specific parking needs. Parkalot emphasizes reservation and allocation, Flexi Parking focuses on convenient digital payment and session management, and JomParking targets quick payment and notifications. These distinctions provide insight into current solutions and inform the design considerations for developing a parking reservation and management system tailored for the UTAR Kampar campus which requires a combination of reservation capability, user convenience, and real-time management.

CHAPTER 3

System Methodology/Approach

3.1 System Approach

To address the parking challenges at UTAR Kampar campus, this project implements a credit-based smart parking system that relies on reservation data to manage parking availability rather than implementing costly IoT sensor infrastructure. The solution centers on a mobile application designed to enable end users to book parking lots in advance through a credit-based management system designed to ensure fair use of parking resources among all campus users.

The system incorporates several key features including real-time parking availability display that allows users to view current empty slots across campus zones, an intelligent parking recommendation system that suggests optimal slots based on user requirements, and a reservation system for booking parking spaces using allocated credits. To enhance user experience, the application provides GPS navigation assistance that guides users to their reserved parking spots through route guidance, eliminating the uncertainty of locating assigned spaces and reducing time spent searching for parking.

To enforce fair use policies and prevent system misuse, the system incorporates penalty mechanisms for reservation violations such as no-shows and inappropriate cancellations. The credit-based framework ensures equitable access by providing fixed monthly credit allocations to users, preventing the monopolization of parking resources while encouraging responsible usage through structured rules and accountability measures. This integrated approach aims to streamline parking management, significantly reduce traffic congestion during peak hours, minimize the searching time for available spaces, and enhance the overall user experience by combining advanced technology with effective resource management policies.

The major deliverables consist of an operating mobile app which would be available on both Android and iOS platforms, which will serve as the interface where users interact with the parking management system. The system features include real-time parking slot availability display, slot recommendations, booking, navigation, and a credit management system.

In the actual implementation, the system will feature real-time availability detection using infrared (IR) sensors embedded in parking spots. These sensors will be connected to the firebase via a microcontroller (ESP32), transmitting data on available parking slots directly to the mobile app. Figure 3.1.1 illustrates the data flow between IoT hardware and the Firebase backend.

Figure 3.1.1 IoT Data Flow

However, for the purpose of this project, and considering both the defined scope and hardware cost limitations, the parking slot status across different zones will be simulated. Instead of relying on sensor-based detection, slot availability will be derived from the reservation records maintained in the database.

Users can reserve parking spaces in advance via mobile device. A credit-based system will manage these reservations in which users can use parking credits for booking spots. With rules and penalties being set to deter misuse, users will lose credit if they fail to utilize or cancel their reservations within a specified period. At every reserved parking slot IoT enabled barriers controlled (Figure 3.1.2) through a mobile app would be installed to prevent unauthorized car parking. Users would unlock these barriers by scanning the QR code upon arrival to ensure that only the rightful individual occupies the reserved space. This hard control approach helps guarantee proper usage of parking resources. However, similar to the IR sensors, installing a physical barrier at each parking slot would lead to high hardware costs. Therefore, the implementation of these barriers is excluded from the current project scope.

Figure 3.1.2 Remote control parking lot barrier

Google Maps is integrated into this project to provide turn-by-turn navigation for users to reach their reserved slot. This feature reduces the search time for parking and helps alleviate traffic congestion in the campus area. Additionally, an intelligent recommendation system helps users make decisions by suggesting the optimal parking spots based on the user's destination, total required time and the availability of empty slots across different parking areas.

3.2 System Design Diagram

3.2.1 Use Case Diagram

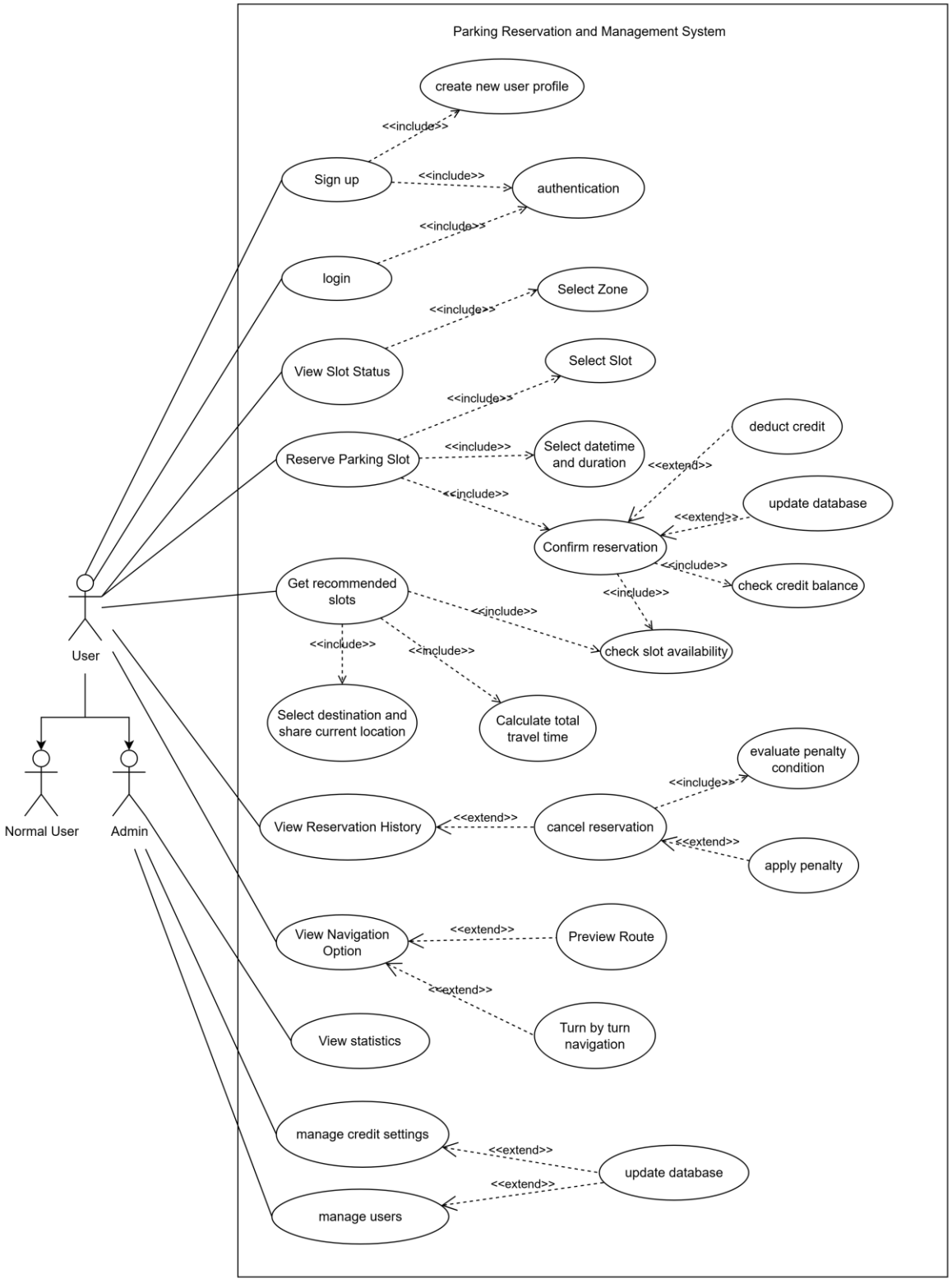


Figure 3.2. 1 Use Case Diagram

3.2.2 Use Case Description

a. View Slot Status

Table 3.3: Use Case Description for Viewing Slot Status

Use Case Name: View Slot Status	Actor: User	ID: 1
Description: - The use case outlines the process of how user view available parking slots in each parking zone through the mobile app. The system will present the status of all slots using color-coded indicators		
Trigger: The user navigates to the reservation interface and selects a specific zone. Type: External		
Relationships: - Association: Users interact with the system to check the availability of parking slots. - Include: Select parking zone - Extend: -		
Normal Flow of Events: 1. User navigates to the reservation interface 2. Users select a parking zone. 3. The system displays all parking slots of the selected zone, and each slot is display with a color-coded status indicator.		
Alternate/Exceptional Flows: -		

b. Reserve Parking Slot

Table 3.4: Use Case Description for Reserve Parking Slot

Use Case Name: Reserve Parking Slot	Actor: User	ID: 2
Description: - The use case outlines the process of how user reserve an available parking space.		
Trigger: The user selects a parking slot manually or clicks on a system-recommended slot. Type: External		
Relationships: - Association: Users interact with the system to initiate a parking reservation. - Include: Select parking slot, select datetime and duration, confirm reservation - Extend: -		
Normal Flow of Events: 1. User navigates to the reservation interface 2. If the user chooses to reserve a slot manually (instead of using the system recommendation), they must select a parking zone. 3. User selects a parking lot. 4. User selects the desired reservation date and duration 5. User click the reserve button. 6. The system will inspect the status of selected slot. 7. The system checks if the user has enough credit balance. 8. If the two conditions are met, the reservation is confirmed successfully; otherwise, the reservation is rejected. 9. The system deducts the required credits from user credit balance. 10. The system updates the reservation data and set the slot status to “Reserved” 11. Display reservation confirmation message.		
Alternate/Exceptional Flows: 1. Insufficient Credit - Display error message 2. Slot unavailable - Display error message		

c. Get Recommended Slot

Table 3.5: Use Case Description for Get Recommended Slot

Use Case Name: Get Recommended Slot	Actor: User	ID: 3
Description: - The use case outlines the process of how user get system recommendations for parking slots based on their selected starting point and destination.		
Trigger: The user selects their destination and chooses their starting point (or current location). Type: External		
Relationships: - Association: Users interact with the system to get parking slot recommendations. - Include: Select destination, select starting point, check slot availability, calculate total time - Extend: -		
Normal Flow of Events: 1. User navigates to the reservation interface. 2. User selects the destination building from a predefined list. 3. User selects the starting point or chooses to use their current location. 4. The system will calculate the estimated driving time according on the starting point to each parking zone. 5. The system checks the walking time from each parking zone to the destination building based on predefined walking data. 6. The system calculates the total time for each zone by combining the driving and walking times. 7. The system will select the parking zone with the shortest total time and check the availability of all the slots within the zone. 8. The system recommends the top 5 available parking slots to the user. 9. Display the 5 recommended parking slots in a list.		
Alternate/Exceptional Flows: -		

d. View Navigation Options

Table 3.6: Use Case Description for View Navigation Options

Use Case Name: View Navigation Options	Actor: User	ID: 4
Description: - The use case outlines the process of how user navigate to their reserved slot.		
Trigger: The user clicks on the navigation button of the active reservation in the list. Type: External		
Relationships: - Association: Users interact with the system to navigate to their reserved slot. - Include: - - Extend: preview route, turn by turn navigation		
Normal Flow of Events: 1. User navigates to the My Reservation interface. 2. User clicks on the navigation button next to an active reservation in the list. 3. The system request for location permission. 4. User selects a navigation option. Option 1: Preview Route i. The system displays an embedded Google Map showing the route to the reserved slot. ii. Users can choose to view step-by-step directions. Option 2: Turn by Turn Navigation i. The system redirects the user to the Google Maps with the destination preloaded. ii. Turn-by-turn navigation is started in Google Maps.		
Alternate/Exceptional Flows: 1. Location Permission Denied - The system displays error messages.		

3.3 System Architecture Diagram

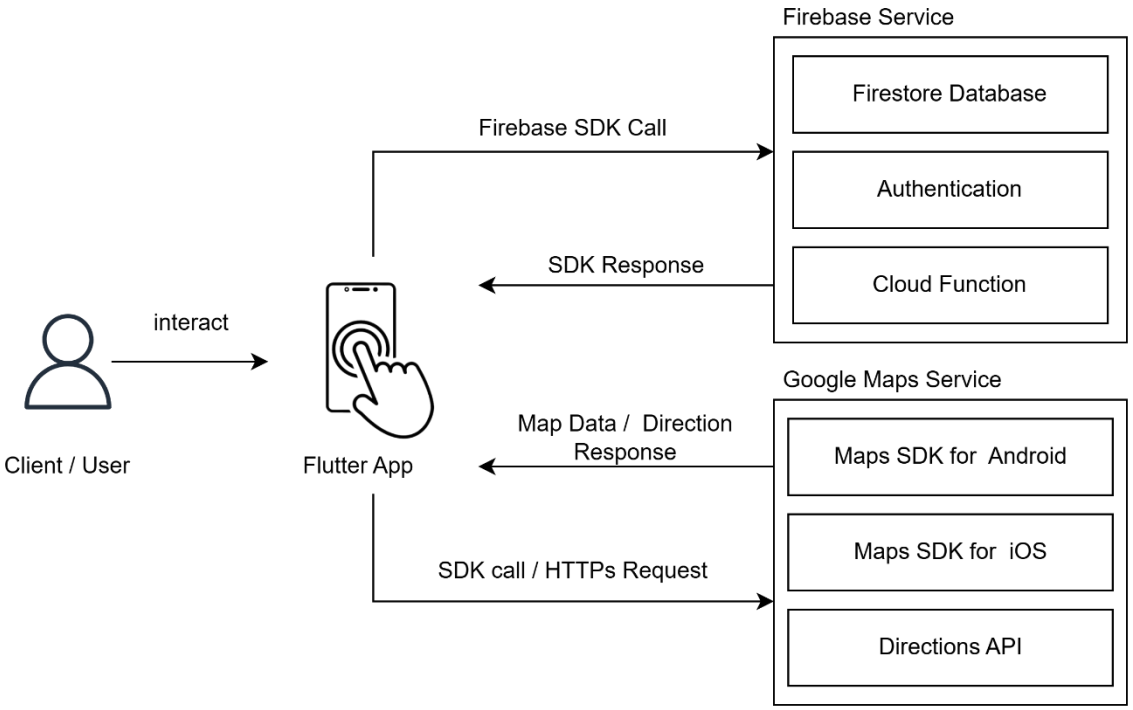


Figure 3.3 Client-Server Architecture Diagram

The diagram above illustrates the client-server architecture for the parking reservation and management system. It shows how users interact with the various services integrated into the mobile application. In this project, both Firebase and Google Maps services are integrated to support various features that enhance the user experience and operational functionality. The Firebase services used in this project include Firestore, Firebase authentication, and cloud functions. Firestore database is used for storing all app-related data and important records such as user details, reservation information, and the locations of campus infrastructure like buildings, parking zones, and parking slots. This ensures real-time data updates and synchronization across all users and devices.

Firebase Authentication is used to verify users when they log in or sign up. It provides different sign-in methods namely the google sign in, phone number and email or password. When a user tries to log in or sign up, the app uses the Firebase Authentication SDK to send a request to the Firebase Authentication service. Firebase then verifies the user's credentials or token and sends

a response back to the app. After receiving the response, the app stores the Firebase ID token and refresh token to maintain the user's session and allow them resigned in after logging out. Cloud Function is crucial to handle custom backend logic such as updating reservation statuses, reallocating credits, and the parking slot recommendation algorithm. When a user interacts with the app or when scheduled events occur, the app triggers a cloud function through HTTP request or service event. The cloud function runs the defined logic such as identifying the optimal parking slot or updating the reservation and slot status after the reserved time has ended. Then it will send a response back to the app to indicate success or return an error message which depends on the outcome of the custom logic defined.

Besides that, google map services are used in both frontend and backend. The google map service included Maps SDK for android and iOS, and Direction API. In frontend, the maps SDK for both android and iOS is used for map display, zooming, and location marking while direction API is used for in-app route preview and generating turn-by-turn navigation which redirect user to google map app for guidance. For backend, directions API are used to calculate the driving time from the user's location to their selected destination. The app will send HTTP request or SDK scall to google map service, then google map will respond with the requested map data or directions information to support the navigation and slot recommendation features.

3.4 Project Timeline

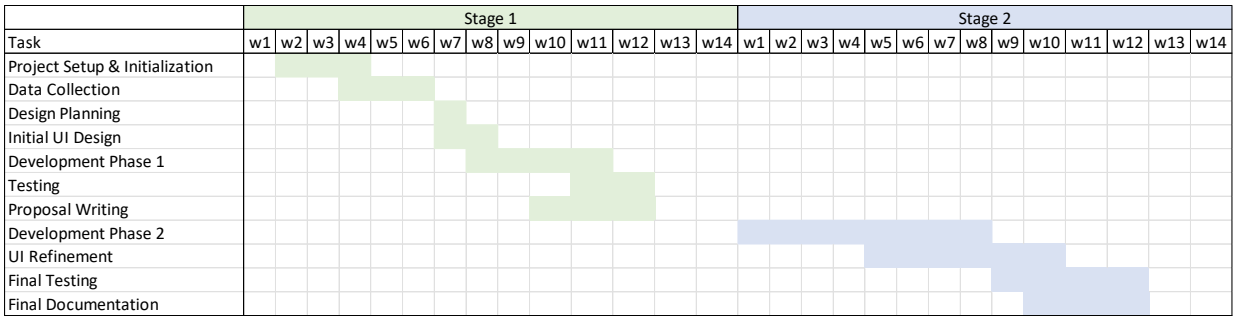


Figure 3.4 Project Timeline

Figure 3.4 shows the timeline and tasks completed in this project. The project is divided into 2 stages, where Stage 1 corresponds to February Trimester 2025 (FYP1*)* and Stage 2 corresponds to June Trimester 2025 *(FYP2)*.

Stage 1 starts with project setup and initialization which includes environment setup, project file creation, linking the frontend and backend projects to Firebase, and installing the required tools or software. After that, collect the important data for the navigation features which is the campus infrastructure location data (buildings, parking zones, and parking slots). The next step is design planning and user interface (UI) design for the mobile application. This is followed by development phase 1 in week 8, this includes the reservation workflow, the parking slot recommendation algorithm, navigation integration using Google Maps services, and user authentication (sign-up and login). After completing development phase 1, the app testing will be conducted to ensure smooth demonstration flow, identify bugs, and determine tasks to be proceeded in development phase 2. Simultaneously, the proposal writing for this project will start in week 10 and continue until week 12.

Stage 2 starts with an eight-week development phase aimed at completing all features in the app and continuing any unfinished tasks from Phase 1. UI refinement will begin in Week 5 and will proceed concurrently with ongoing development. The UI is updated to ensure a smooth and efficient workflow. Final testing will start in week 9, after the complete development phase 2. This step is important to ensure that all the features are completed to ensure they can work without errors and all the system features perform as expected. Finally, the project ends with the preparation of final documentation, covering all the work completed throughout the project.

CHAPTER 4

System Design

Building upon the system methodology and design discussed in Chapter 3, this chapter explains how the system operates in practice. It describes the flow of user interactions with the application and highlights how each module works together to achieve the intended functionality. In addition, it also provides detailed specifications, designs of the system components and their respective roles in supporting the application.

4.1 System Flowchart

4.1.1 User Authentication flow diagram

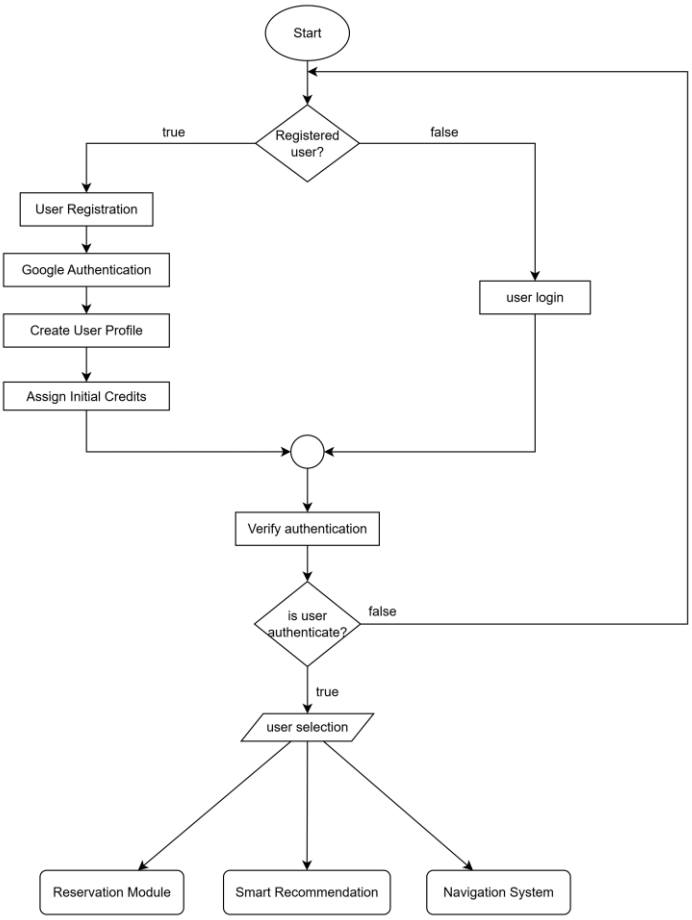


Figure 4.1.1 System flow diagram

The system flow diagram provides an overview of how the end users interact with the parking reservation application. The system begins with user authentication by checking their registration status. New users proceed through Google authentication and profile creation, while existing users go through the login verification process. Once authenticated, users can access the three core modules: Reservation System, Smart Recommendation System, and Navigation System.

4.1.2 Parking Space Recommendation System Flow

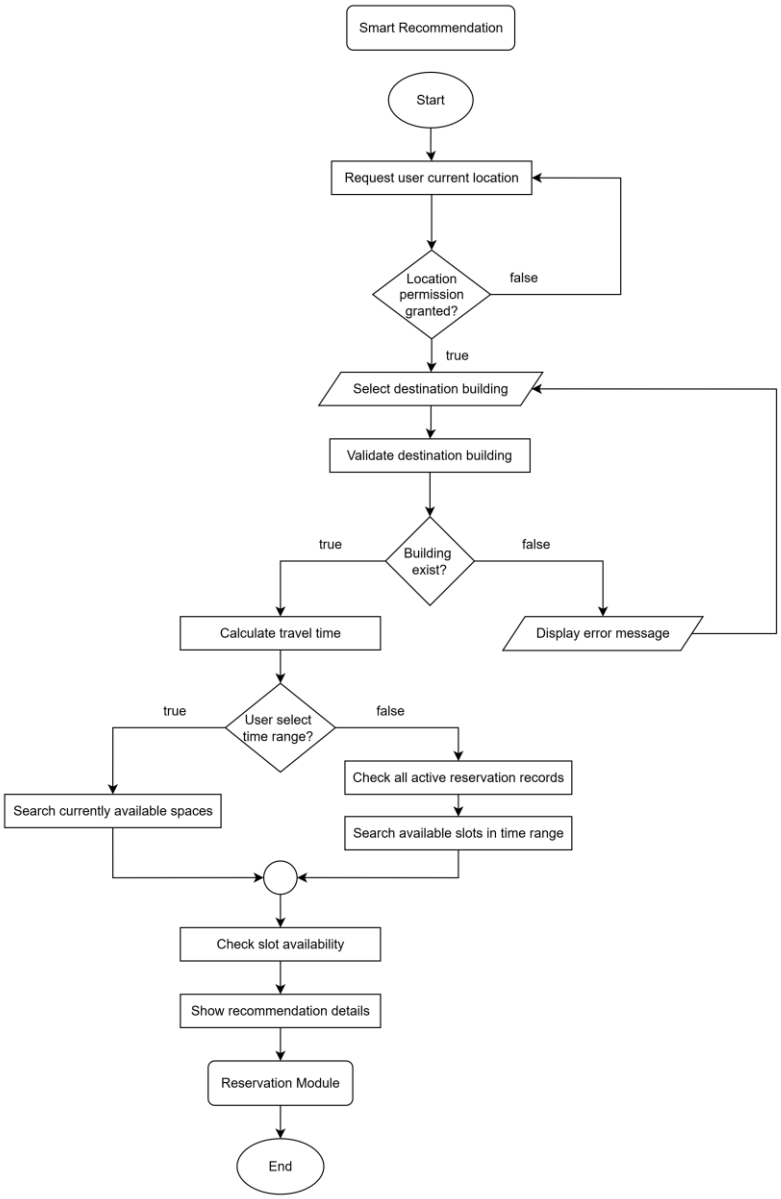


Figure 4.1.2 Smart recommendation flow diagram

The Smart Recommendation Module is designed to recommend suitable parking slots to users based on the total travel time to their selected destination. To use this feature, the user must grant location access permission. The recommendation process begins when the user selects the destination building and optionally specifies a preferred time range. The system then calculates the estimated travel time and identifies available parking spaces. If no time range is specified, the system searches for slots that are currently available. If a time range is provided, the system cross-checks active reservation records to determine slots available during the specified period. The system then verifies slot availability and presents recommended options to the user. From these recommendations, the user may proceed to make a reservation.

4.1.3 Reservation Flow

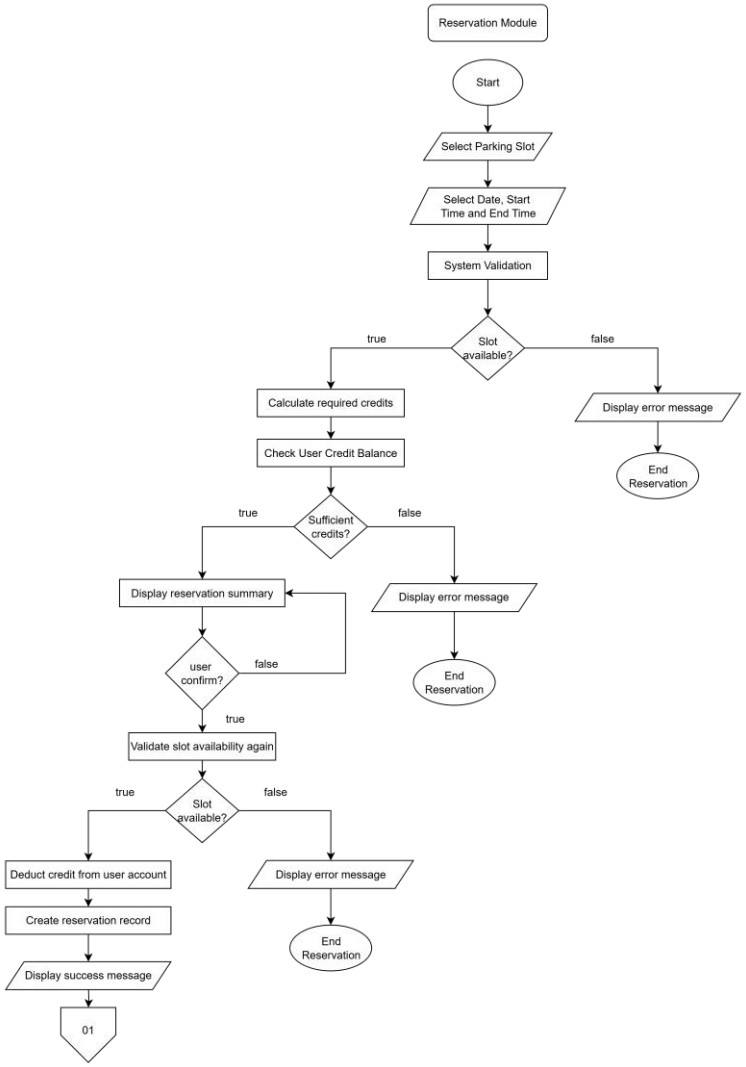


Figure 4.1.3 Reservation flow diagram

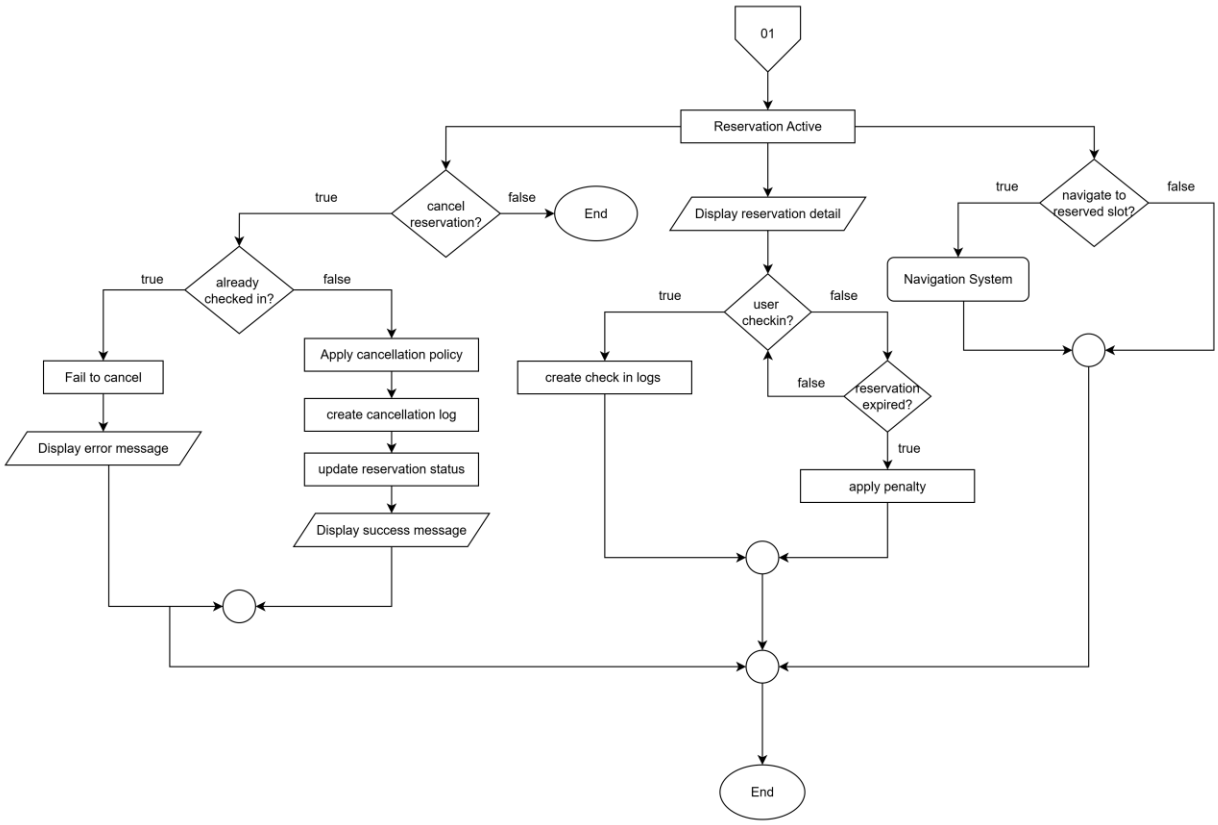


Figure 4.1.4 Reservation flow diagram

Figures 4.1.3 and 4.1.4 illustrate the system flow of the reservation module. To initiate a reservation, the user selects the desired parking slot and specifies the intended date and time range. If the slot is available, the system verified the user's input and calculated the number of credits required. Once the user has sufficient credits and confirms the reservation, the system re-validates the slot availability to ensure it has not been booked by another user. Upon successful validation, a new reservation record is created, and its status is set to active. An active reservation can be managed by the user in several ways. First, the user may cancel the reservation before check-in. In this case, the cancellation policy will be applied, and the percentage of refunded credits depends on the timing of the cancellation. The system records this action in a cancellation log and updates the reservation status accordingly. Second, the user may check in before the reservation expires. The system records this action in a check-in log. If the user fails to check in within the reservation period, penalty will be applied. For example, deducting credits or reducing the value of the monthly credit allocation. Finally, the system also allows users with active reservations to navigate to their reserved slot via the navigation module.

4.1.4 Navigation System Flow

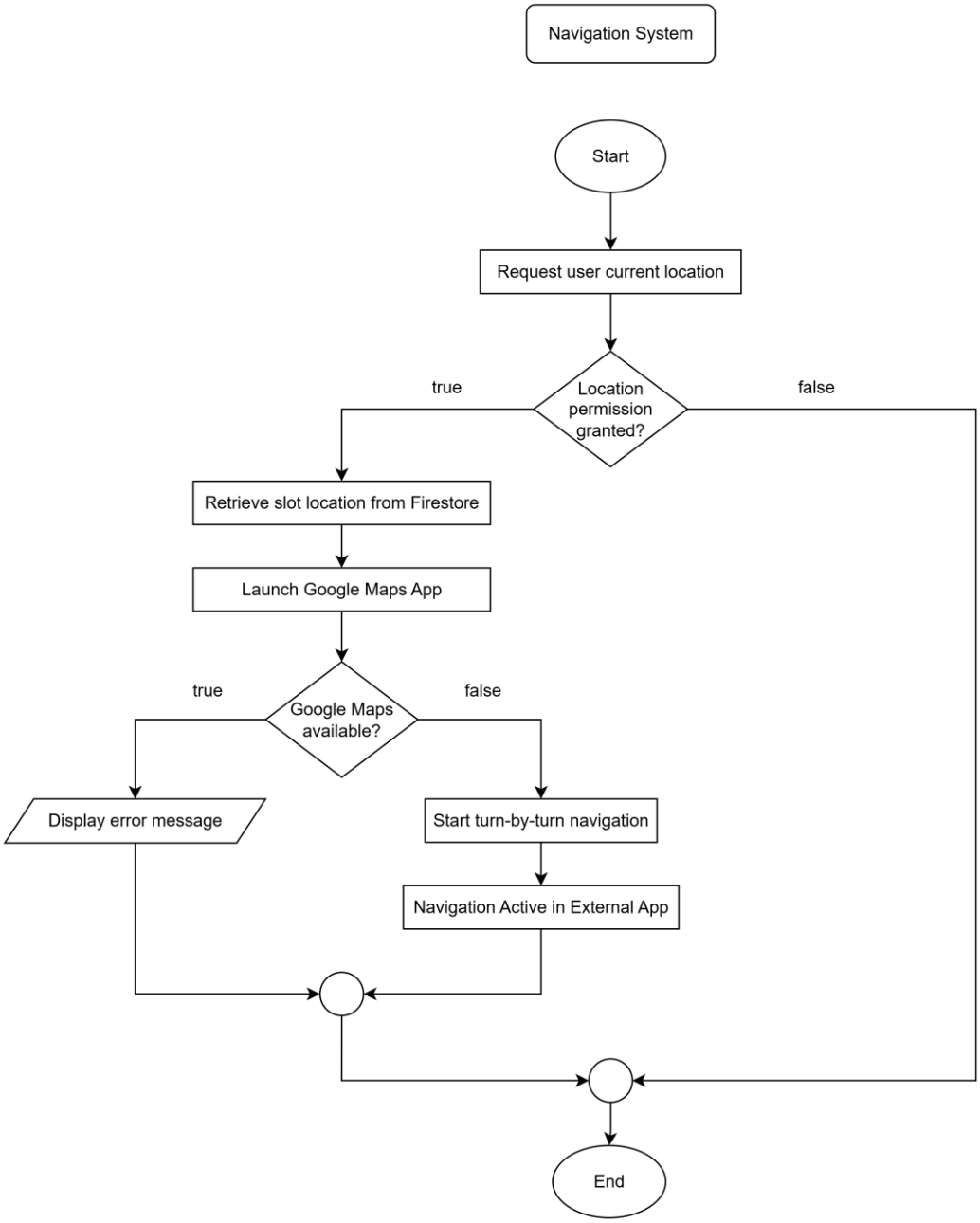


Figure 4.1.4 Navigation flow diagram

The navigation flow diagram in Figure 4.1.4 shows the process by which a user can navigate to their reserved parking slot. To initiate navigation, the user must share their current location. The system then retrieves the corresponding parking space location data from the Firestore database and subsequently launches Google Maps to provide turn-by-turn navigation.

4.2 System Components Specifications

4.2.1 Mobile Application Platform Specifications

Table 4.2.1 Mobile Application Platform Specifications

Component	Specifications
Platform	Flutter
Flutter version	3.29.3
Dart SDK version	3.7.2
Flutter Channel	Stable
Target platforms	Android: API 23+ (Android 6.0+)
	iOS: 12.0+
IDE	Android Studio 2022.1+, VS Code

4.2.2 Backend Platform Specifications

Firebase as a Backend-as-a-Service (BaaS) platform was used to handle backend logic more efficiently without the need to maintain a dedicated server or database. It supports cross-platform mobile app development and provides various services that align with the project requirements.

For user authentication, the **Firebase Authentication** service supports multiple sign-in methods. For instance, Google sign-in, email and password, Apple ID, and phone number. This service ensures that only verified and authenticated users are granted access to the application features. Through the authentication service, user information is securely stored in the cloud and remains consistently accessible across different devices.

One of the core services used is **cloud firestore**. Firebase Firestore is a NoSQL document-oriented database that offers flexibility, scalability to handle large number of concurrent users, and real-time synchronization, which is crucial for updating parking space status in the application. It is responsible for managing all application data, including user details, reservation records, location-based data, credit configuration settings, and various logs. The

database supports real-time updates, enabling instant synchronization of slot availability. In addition, its offline persistence feature allows users to continue interacting with the system even when temporarily disconnected from the internet, with changes automatically synchronized once the connection is restored.

Another critical component is Firebase **Cloud Functions**. This service is essential to host and execute backend logic in the cloud without the need to manage physical servers. In this project, cloud functions handle operations such as creating and cancelling reservations, automatically updating slot statuses based on reservation times, and managing monthly credit allocations. They also crucial in supporting the credit system by enforcing credit rules and applying penalties to prevent misuse of the system. By running these functions on Firebase's serverless infrastructure, the application ensures secure execution of backend processes without depending on local machines.

4.2.3 Third-Party Service Specifications

In this project, the Google Maps service is used to provide location-based functionalities. It supports seamless map integration within the application which enables users to check available parking spaces and make selections before making a reservation. The service also displays the exact location of each parking space within the campus so that users can have a clear understanding of where their reserved space is situated. Google Maps assists users in navigating to their reserved slot by providing accurate directions and turn-by-turn navigation.

4.2.4 Hardware and Software Requirements

The hardware used in this project includes laptop and mobile devices. Laptops are used for development while mobile devices are used for testing purposes.

Table 4.2.4.1 Laptop Specifications

Description	Specifications
Model	Asus Vivobook 15
Processor	Intel Core i5-1135G7
Operating System	Windows 11
Memory	12GB RAM
Storage	500GB SSD, 480GB HDD
Network	Internet connection required

Table 4.2.4.2 Testing Device

Description	Specifications
Emulator/Simulator	Android Emulator / iOS Simulator
Supported OS Versions	Android 6.0+ or iOS 12.0+
GPS	Supported
Internet connectivity	Required
Storage	1 GB free space

4.2.5 Development and Design Software

One of the software tools required for development is Visual Studio Code (VS Code). It is the code editor for building the Flutter application as well as writing backend Cloud Functions in JavaScript. VS Code was chosen because of its lightweight nature, highly customizable, and supports a wide range of extensions. For example, flutter and firebase plugins for debugging and testing.

Flutter and Dart SDK are the cross-platform mobile application framework used in this project. This software enables the app to run seamlessly on iOS and Android devices. The dart programming language is used for flutter development to build responsive user interfaces and implement application logic. Through official firebase plugins, flutter can establish connections with Firebase services. The framework ensures consistency in design and functionality across platforms while reducing development effort and maintenance overhead.

4.3 System Components Design

4.3.1 Mobile Application Component Design

The Flutter mobile application follows modular architecture with five main components. The first component is the model; it is used to define the data structures used across the application. They provide the foundation for data handling and ensure consistency between the Firestore database and the user interface. The second component is the pages which serve as the primary structure of the application. It is organized into four main modules, each containing its own subpages to handle different functions. All these pages are grouped within a single folder to maintain a clear and consistent project structure.

Additionally, the screens component manages navigation-related views and integrates with Google Maps to guide users to their reserved parking slots. The widgets component contains reusable user interface elements that keep the design consistent and help reduce duplication. The last component is the services. Services components provide business logic and system functions. They handle CRUD operations and interact with Firebase Cloud Functions to complete system tasks and process user requests. This modular design makes the application easier to maintain and scale while keeping user and administrative functions well organized.

The mobile app is organized into four main modules that are accessed through a bottom navigation bar: Reservation, Reservation History, Profile and Setting. Each module contains several subpages. The reservation module displays real-time parking slot availability by marking each zone and slot on a google map. The color of the marker indicates the status of the slot. Users can manually select the desired parking space on map. It also provides a recommendation subpage where users can receive system-generated suggestions before proceeding with a reservation. The reservation history module presents all reservation histories, including active, completed, and cancelled reservations. This module also includes subpages for viewing reservation details, performing check-in through a QR scanning screen, and accessing navigation. The profile module displays the user's account information and credit balance. It also provides access to a credit transaction history subpage, allowing users to review all credit usage.

The Settings module is accessible only to administrators and provides essential tools for managing the system. Administrators can monitor important statistics such as the number of active users in the last 30 days, today's total reservations, cancelled booking and slot occupancy rates. This module consists of two subpages: Credit Management and User Management. The Credit Management subpage allows administrators to view and update credit configurations, including the monthly credit allocation, credit rates and durations, as well as penalty values for no-shows and cancellations. The User Management subpage displays all registered users along with their associated information, and administrators can update user details when necessary to maintain accuracy and ensure smooth system operation.

4.3.2 Core System Components Design

Real-Time Parking Space Status Display Design

The application is designed to display parking space status in real time. It allows users to check slot availability before making a reservation. Live data streams and scheduled maintenance processes are used to maintain accurate availability across multiple zones. Firebase Firestore listeners are used to provide instant updates whenever a slot's status changes to ensure that users can view the latest availability without the need to refresh the app manually. Within the application, Google Maps is integrated to visually represent zones and slots. Each slot is marked at its specific coordinates, with marker colors indicating the status. To ensure further accuracy, a scheduled cloud function runs periodically to update slot statuses based on reservation records and check-in logs. This process guarantees that availability data remains reliable to support informed user decisions when selecting a parking space. Additionally, whenever an individual slot's status changes, the parent zone's available count is updated automatically to reflect the number of empty slots in that zone.

Reservation and Credit System Design

The credit system is a core part of the mobile application. It is designed to ensure fair access to limited parking spaces and prevents misuse of the reservation feature. The system sets rules for penalizing no-shows and last-minute cancellations. All credit-related configurations should be stored in the database and can be updated by the administrator when necessary. In addition, all the credit transactions are recorded in the database, users can review their credit usage history directly in the app. The credit value is defined as one credit for each hour of parking reservation. One credit equals one hour of parking reservation. Credits cannot be transferred between users; they are reset at the beginning of every month and are not allowed to be carried forward. Advance booking is allowed only within the current month, so credits from this month cannot be used for booking spaces for the next month. This design encourages efficient use of campus parking since every user has a limited chance to reserve a slot and must plan their usage carefully.

The cancellation policy is designed to balance user flexibility with fair use of parking resources. Users may cancel the reservation within five minutes of booking and receive a full credit refund. Cancellations are not allowed within 30 minutes of the reservation's end time. Before

the reservation period begins, time-based cancellation rules are applied to determine the percentage of credits refunded. Users who cancel earlier are eligible for higher refunds while last-minute cancellations or those made during the reservation period result in reduced refunds and penalty credits. Credit refunds are introduced to encourage users to cancel reservations they no longer need so that the parking space is available for others. Penalties are applied to users who block parking resources by reserving a slot without using it. Together, the refund and penalty mechanisms ensure that users are motivated to act responsibly and that parking spaces are utilized efficiently.

The no-show penalty system is designed to discourage users from reserving a parking space without using it, as this prevents others from accessing the slot and leads to wasted parking resources. A no-show penalty is applied when the system detects an expired reservation without a check-in record. The system uses progressive penalties that increase with repeated offenses. A light penalty is applied as a warning for first-time offenders, while repeated violations result in higher credit deductions and may even reduce the user's monthly credit allocation. All offenses are tracked over time, and penalties are applied automatically to maintain fairness and prevent abuse of the reservation system.

Smart Recommendation Algorithm Design

The recommendation module is designed to suggest the most suitable parking spaces for users based on travel efficiency and availability. The algorithm considers two main factors: the driving time from the user's location to each parking zone and the walking time from the parking zone to the selected destination building. These two values are combined to identify the zone with the least total travel time. After receiving the user's GPS coordinate and destination, the system retrieves real-time zone and slot information from the database. The algorithm relies on the Google Maps Directions API to calculate the estimated driving time and uses pre-stored walking times between zones and campus buildings to determine the additional time needed to reach the destination. By summing up these values, the total travel time for each zone is obtained. The system then should check slot availability within each zone. If the user specifies a time range, the system validates whether slots remain unoccupied during that period. Finally, the top five results are returned to the user, ensuring recommendations that minimize travel effort while reflecting real-time slot availability.

4.3.3 Database Design Architecture

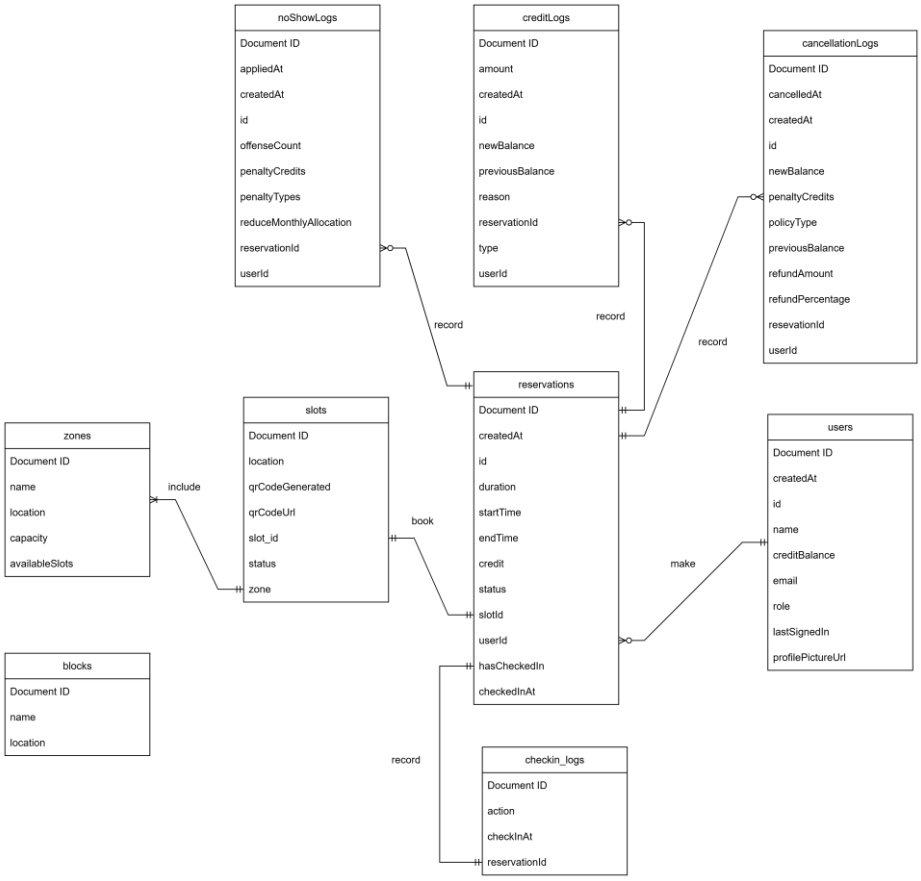


Figure 4.3.3.1 Document Relationship Diagram

In this project, Firestore is used as a NoSQL document database to manage all application data. To provide a clear overview of the data schema, a document relationship diagram is presented in Figure 4.3.3.1. Tables are used to represent each collection, its documents, mappings, and relationships. Three key collections related to campus infrastructure location data are blocks, zones, and slots. The blocks collection stores the coordinates of campus buildings in UTAR Kampar which are essential for destination selection and navigation. The zones collection contains information about parking areas, including their coordinates, total capacity, and available slot count. The slots collection represents individual parking spaces, each belonging to a specific zone. For performance and efficient querying, slots are not stored as subcollections under zones but are instead maintained in a separate collection. Each slot document stores its coordinates for navigation, status for real-time availability display, and a QR code URL for check-in purposes.

User information is stored in the user's collection which includes fields such as name, email, and role. The role field determines each user's level of access to system features. The last sign-in time is recorded for statistical purposes, while the profile picture URL is stored to display the user's profile image. The credit balance serves as the currency for making parking reservations on campus and is automatically reset each month to ensure fair access to parking resources. The reservations collection stores essential details for each booking, including creation time, duration, start time, end time, credits used, status, slot id, and user id. When making a reservation, the user selects the desired parking slot and specifies the intended time range. The reservation status indicates whether the booking is active, canceled, or completed. The user id field records which users created the reservation, while the slot id links the reservation to a specific parking space.

In addition to reservations, several log collections are linked to the reservations collection to track user actions and events. The check-in log records when a user checks in to their reserved parking space by capturing the QR code. It stores the user action together with the check-in time. The cancellation log captures details of reservation cancellations. It records the cancellation time and keeps a snapshot of the user's credit balance by storing both the previous and the new balance. It also includes information about penalties for late cancellations as well as the refund amount and percentage. The no-show log is used to record situations where a user fails to check in for their reserved slot. It stores details of the penalty such as the base penalty credit, the offense count, the final penalty credit, and the penalty type. For repeated offenses, the system reduces the user's monthly credit allocation, and the reduced amount is stored in the log as well. Furthermore, the credit log records all credit transactions, including credits spent on reservations and credit refunds to users when cancellations are made. For all log collections, a user reference is stored to enable faster and more efficient queries.

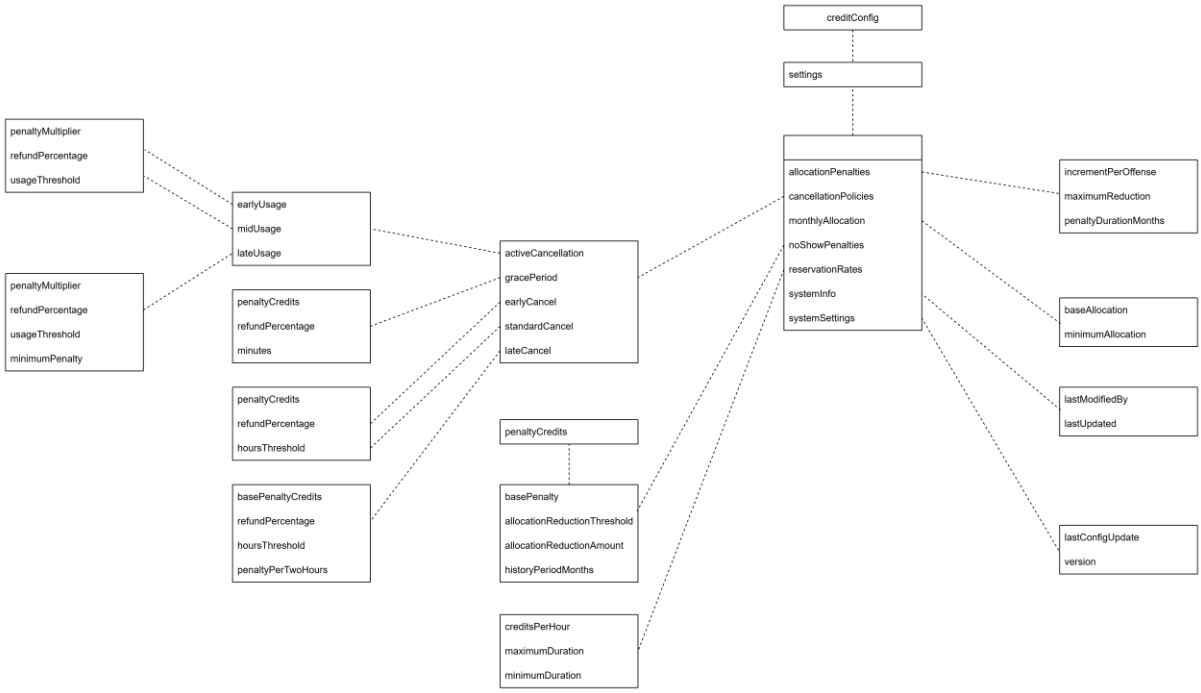


Figure 4.3.3.2 Credit Setting Document Relationship Diagram

Figure 4.3.3.2 demonstrates the document relationship diagram for the credit system configuration. The credit system data is stored in the database to enable administrators to update values through the application interface without requiring code modifications. The creditConfig collection serves as the central configuration repository. It includes the system-wide settings that control all aspects of credit management behavior. The settings document consists of several key policy areas, including allocation penalties, cancellation policies, monthly allocation settings, no-show penalties, reservation rates, system information, and system settings. This centralized approach ensures consistent policy application across the entire system while maintaining flexibility for administrative adjustments.

The allocation penalties configuration defines parameters such as incremental penalties per offense, maximum reduction limits, and penalty duration. These penalties are applied to users with repeated unused reservations. The no-show penalties configuration is designed as a progressive system that escalates based on records stored in the no-show log, as illustrated in Figure 4.3.3.1. The configuration has 2 main components which are base penalty and scaling settings. The base penalty refers to the penalty credits applied to offenders. The historical period that determines how long past offenses remain valid. Threshold is used to determine

when allocation reductions are triggered and the amount that define the deduction credit value for allocation. This structure provides administrators with the flexibility to configure penalty severity and escalation to ensure both fairness to users and consistency in handling no-show cases.

The cancellation policies are used to handle user cancellations under different conditions. They are divided into four tiers: early, standard, late, and active cancellations. The refund amount depends on the time when the user performs the cancellation, and penalties may be applied if the cancellation occurs during the reservation period. Each tier defines parameters such as penalty credits, refund percentage, applicable duration, usage thresholds, and minimum penalty.

For monthly allocation settings, they configure the baseline of the credit distribution system. They specify the amount of credit allocated to users each month, as well as the minimum guaranteed allocation even for users who have been penalized.

Chapter 5

System Implementation

5.1 Software Setup

Before initiating the development process, project initialization and environment set up by installing the necessary software is required.

Table 5.1.1.1 Tools/Software Required for Installation

Tool	Purpose
Visual Studio Code	Code Editor for Flutter and Node.js
Git	Version Control and manage source code
Flutter SDK	Cross-platform mobile applications framework for Android & iOS
Dart SDK	Programming language for Flutter
Node.js	Backend environment for Firebase
CMake	Use for native plugin and setup emulator

5.2 Setting and Configuration

5.2.1 Firebase Project Initialization

Firebase was used as a Backend-as-a-Service (BaaS) platform to handle backend logic more efficiently without the need to maintain a server or database. It supports cross-platform mobile app development, and the services and features provided align closely with the core requirements of this project. For example, firestore database enables real-time parking slot updates to ensure instant synchronization across multiple users and devices. Besides that, firebase authentication supports various sign-in methods for secure and convenient user authentication. Additionally, cloud function services are used to handle the backend logic for updating parking slots status, credit management, slot reservation and parking recommendations.

1. Create a project in Firebase Console and register flutter apps to get the configuration files
2. Link Firebase to Flutter App by setup configuration file and initialize firebase in main.dart.
3. Initialize Firebase functions and Firestore database for backend

5.2.2 Google Maps API setup

Google Maps service is a crucial component for the project as it provides location-based features to support functions in the project such as map rendering, get user current location, route navigation and smart parking recommendation logic.

1. Create a google cloud project
2. Enable the required Google Maps APIs such as Maps SDK for Android, Maps SDK for iOS and Directions API
3. Generate the API key
4. Add API key to frontend and backend

5.3 Data Collection and Storage

Before entering the development phase, data collection is essential to support the logic of the project. The aim of this phase is to collect accurate spatial and logical data about the campus infrastructure, including building, parking zones and all parking slots within each zone.

5.3.1 UTAR Kampar Campus Parking Zone Layout Map

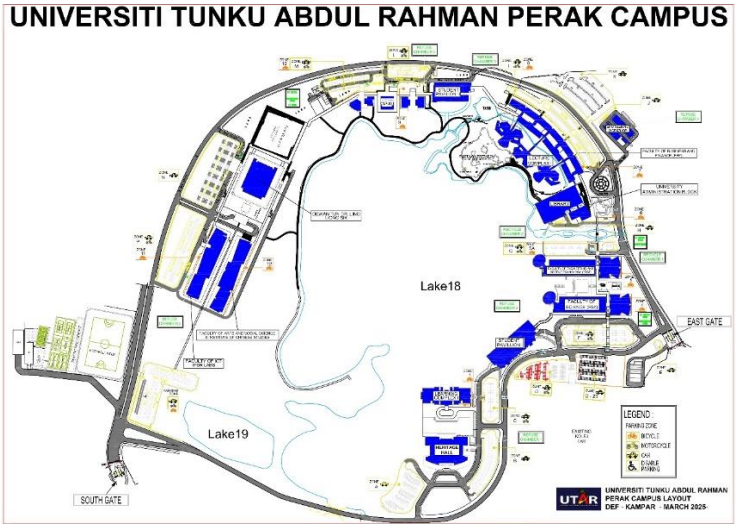


Figure 4.2.1 Parking Zone Layout

5.3.2 Data Collection Approach

GPS coordinate

The GPS coordinates (locations) of the parking blocks and zones are obtained through Google Maps. However, there are some challenges in acquiring GPS coordinates for all individual parking slots. The exact location of each parking space is not provided in Google Maps. Therefore, there is a need to refer to the parking zone layout map to determine the layout and locations on the map. An effort was made to use OpenCV to detect parking space boundaries, but the results were inaccurate due to the small slot size on the map and scale ratio issues. Besides that, the map and the real-life layout are not consistent, the total number of slots shown on the map differs from the actual number. Ultimately, to ensure data accuracy, a manual verification method was chosen but it is slightly time-consuming. Each zone was manually verified by counting the number of slots per row and cross-referenced with the actual campus layout. After confirming and updating the data on the map, the layout was overlaid on a real map using **Scribble Maps** (Figure 4.2.2). Each slot was marked using the marker tool and assigned a unique Id. The map was then exported as a CSV file containing the slotId and the GPS coordinates. This file was used to import the location data into Firestore database.

Figure 4.2.2 Scribble Map with Marked Parking Slot Locations

Walking time

The walking time refers to the time required from each parking zone to each building on campus. The walking time provided by google map is based on the main road, so the data might not be accurate for the actual condition.

Figure 4.2.3 OpenStreetMap

To obtain more accurate data, OpenStreetMap was used to edit the map by adding sidewalks and paths to calculate the distance from each parking zone to the campus buildings. The red dotted line in Figure 4.2.3 indicates the sidewalk. The Valhalla approach was used to identify the shortest path between two different locations as it provides flexible pathfinding based on real-world map data. Walking time is calculated based on an average walking speed of 82 m/min. The shortest distance data is stored in a table in the Excel file, as shown in Figure 4.2.4, and the walking time data is shown in Figure 4.2.5.

		Block														
	Distance (m)	A	B	C	D	E	F	G	H	I	J	K	L	M	N	P
Parking Zone	A	79	191	429	535	625	770	728	1000	853	1100	1300	1200	976	876	876
	B	67	175	317	445	569	659	620	919	997	1000	1152	1233	1100	991	959
	C	143	24	151	280	207	391	490	582	653	576	822	903	1300	1200	1200
	D	404	282	148	180	285	415	523	620	693	721	854	942	1248	1491	1491
	E	569	431	288	98	201	328	433	475	539	576	709	801	1101	1357	1357
	F	478	410	278	35	148	293	392	494	551	586	722	800	1112	1370	1370
	G	591	439	273	169	105	133	101	349	405	454	565	655	1046	1221	1221
	H	475	351	227	150	86	41	146	243	316	358	479	571	860	1137	1134
	I	795	685	554	403	330	210	229	42	77	91	253	345	634	910	910
	J	806	685	541	467	392	300	313	110	177	20	337	539	818	1100	1100
	K	904	785	662	482	422	287	317	103	170	70	331	472	761	1000	1000
	L	836	705	644	588	517	414	419	202	235	463	63	74	376	641	633
	M	992	870	782	714	645	571	549	331	361	453	190	38	267	547	512
	N	1172	1272	1194	1077	1001	896	871	654	686	832	513	377	58	173	244
	P	1185	1285	1380	1200	1084	977	952	777	830	779	636	467	226	57	102
	Q	1265	1365	1460	1280	1164	1057	1032	857	910	859	716	547	306	137	182
	SC	714	820	1000	1200	1200	1300	1300	1164	1217	1269	1023	926	337	219	217

Figure 4.2.4 Shortest distances between parking zones and blocks

		Block														
		Block A	Block B	Block C	Block D	Block E	Block F	Block G	Block H	Block I	Block J	Block K	Block L	Block M	Block N	Block P
Parking Zone	Zone A	0.96	2.33	5.23	6.52	7.62	9.39	8.88	12.20	10.40	13.41	15.85	14.63	11.90	10.68	10.68
	Zone B	0.82	2.13	3.87	5.43	6.94	8.04	7.56	11.21	12.16	12.20	14.05	15.04	13.41	12.09	11.70
	Zone C	1.74	0.29	1.84	3.41	2.52	4.77	5.98	7.10	7.96	7.02	10.02	11.01	15.85	14.63	14.63
	Zone D	4.93	3.44	1.80	2.20	3.48	5.06	6.38	7.56	8.45	8.79	10.41	11.49	15.22	18.18	18.18
	Zone E	6.94	5.26	3.51	1.20	2.45	4.00	5.28	5.79	6.57	7.02	8.65	9.77	13.43	16.55	16.55
	Zone F	5.83	5.00	3.39	0.43	1.80	3.57	4.78	6.02	6.72	7.15	8.80	9.76	13.56	16.71	16.71
	Zone G	7.21	5.35	3.33	2.06	1.28	1.62	1.23	4.26	4.94	5.54	6.89	7.99	12.76	14.89	14.89
	Zone H	5.79	4.28	2.77	1.83	1.05	0.50	1.78	2.96	3.85	4.37	5.84	6.96	10.49	13.87	13.83
	Zone I	9.70	8.35	6.76	4.91	4.02	2.56	2.79	0.51	0.94	1.11	3.09	4.21	7.73	11.10	11.10
	Zone J	9.83	8.35	6.60	5.70	4.78	3.66	3.82	1.34	2.16	0.24	4.11	6.57	9.98	13.41	13.41
	Zone K	11.02	9.57	8.07	5.88	5.15	3.50	3.87	1.26	2.07	0.85	4.04	5.76	9.28	12.20	12.20
	Zone L	10.20	8.60	7.85	7.17	6.30	5.05	5.11	2.46	2.87	5.65	0.77	0.90	4.59	7.82	7.72
	Zone M	12.10	10.61	9.54	8.71	7.87	6.96	6.70	4.04	4.40	5.52	2.32	0.46	3.26	6.67	6.24
	Zone N	14.29	15.51	14.56	13.13	12.21	10.93	10.62	7.98	8.37	10.15	6.26	4.60	0.71	2.11	2.98
	Zone P	14.45	15.67	16.83	14.63	13.22	11.91	11.61	9.48	10.12	9.50	7.76	5.70	2.76	0.70	1.24
	Zone Q	15.43	16.65	17.80	15.61	14.20	12.89	12.59	10.45	11.10	10.48	8.73	6.67	3.73	1.67	2.22
	Zone SC	8.71	10.00	12.20	14.63	14.63	15.85	15.85	14.20	14.84	15.48	12.48	11.29	4.11	2.67	2.65

Figure 4.2.5 Minimum walking time required

5.3.3 Database Structure

After collecting the required campus infrastructure data, the dataset was imported from a CSV file into the Cloud Firestore Database to support core system features such as reservation, slot recommendation, and navigation. In addition, several logs are created to record different user reservation actions, including check-in logs, credit transactions, cancellations, and no-show logs. The database successfully stores all user data, booking records, credit transactions, and parking location information with real-time synchronization across all connected devices. Data integrity and consistency are maintained by Firestore's built-in transaction mechanisms to ensure reliable operations for all system functions.

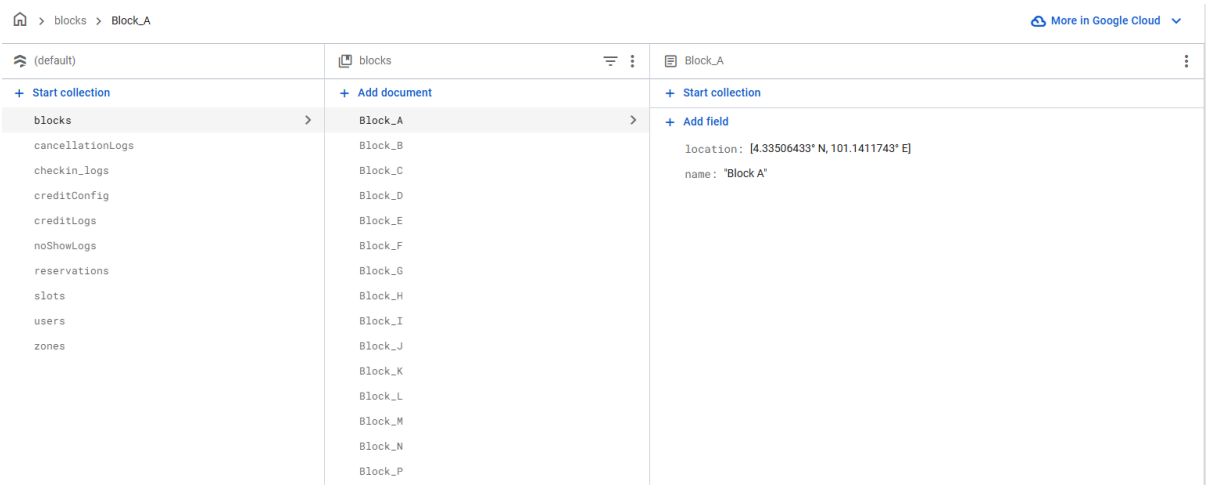


Figure 4.2.6 Cloud Firestore Database

5.4 System Operation

5.4.1 User Authentication Module

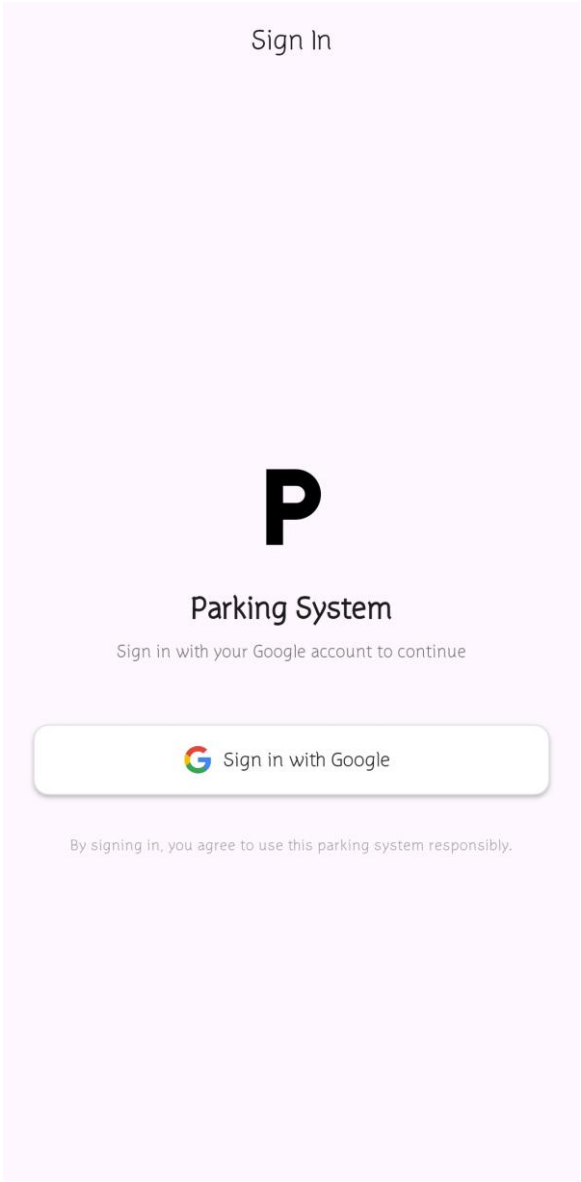


Figure 5.4.1.1 Login Interface

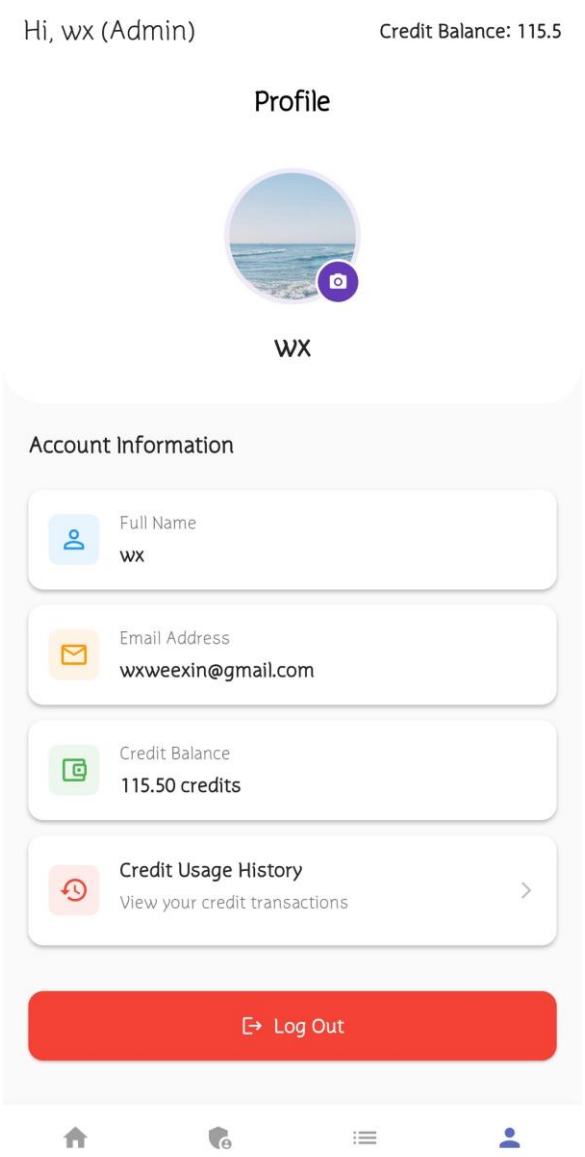


Figure 5.4.1.2 User Profile

Firebase Authentication is used to verify user identities and ensure secure access to the system. Figure 5.4.1.1 shows the login interface. For a smooth and secure login, users are required to authenticate using their Google account. Once authenticated, users are redirected to the home page as presented in Figure 5.4.2. User information is retrieved from Firebase Authentication and synchronized with the Firestore database. The user profile page in figure 5.4.1.2 displays user details such as the name, profile image, current credit balance, and access to view their credit usage. Users can securely end their session using the logout button located at the bottom of the page.

5.4.2 Home Screen

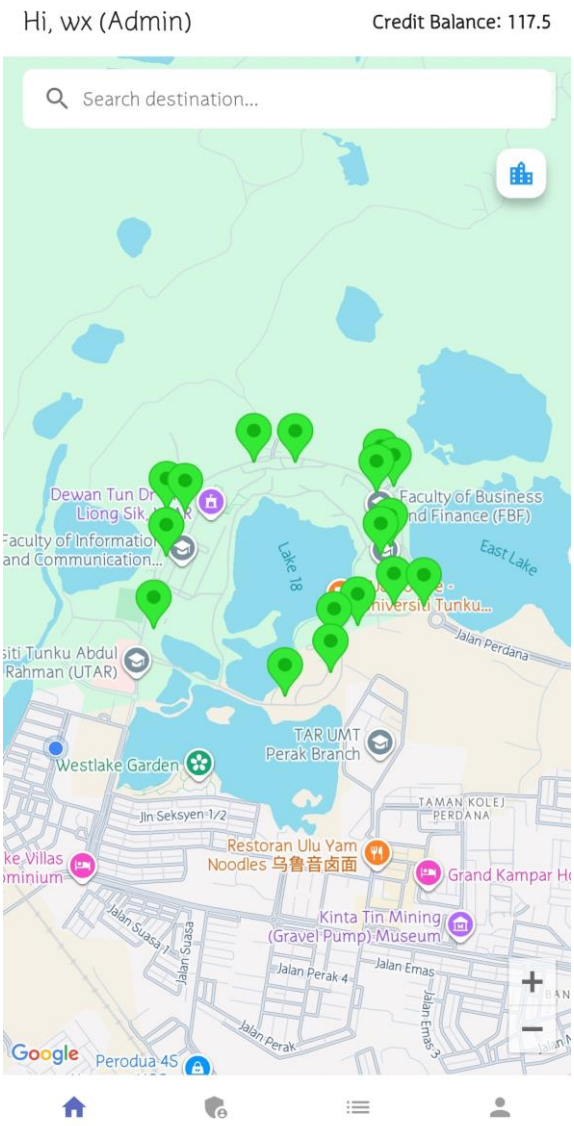


Figure 5.4.2.1 Home page

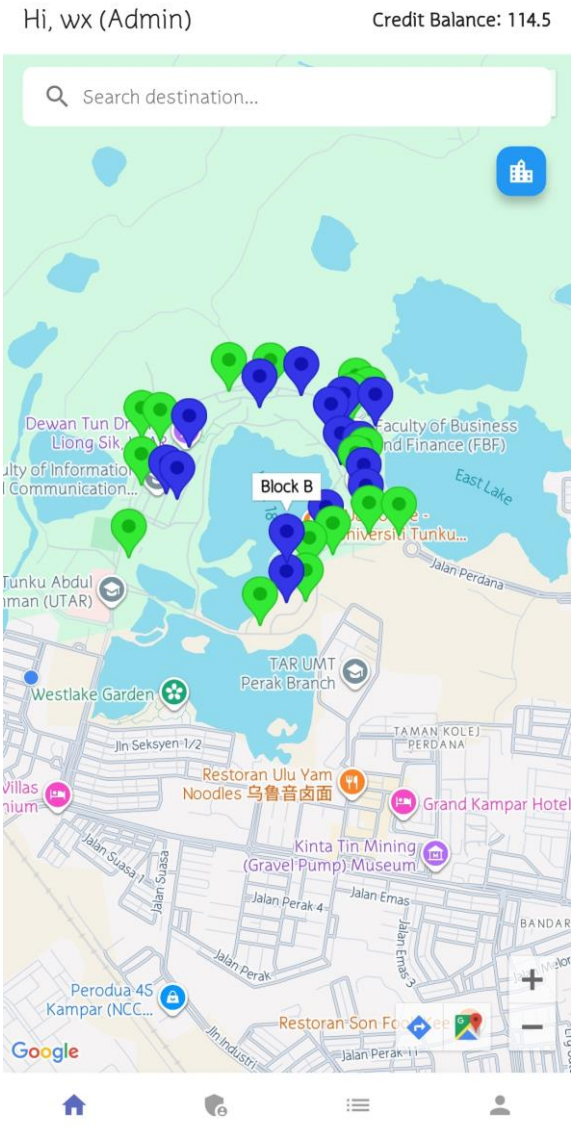


Figure 5.4.2.2 User Profile

On the home page, users are presented with a Google Map that displays the locations of all campus parking areas using markers. Each marker is color-coded to indicate the occupancy status of its corresponding zone. A green marker shows that more than 60% of the slots are available, an orange marker indicates moderate availability, and a grey marker means the zone is fully occupied with no slots left. A search bar located at the top of the screen allows users to request system-generated slot recommendations. Just below the search bar on the right, a toggle button allows users to switch the view to display campus building markers, helping them easily identify their destination.

5.4.3 Real-time Slot Availability Display

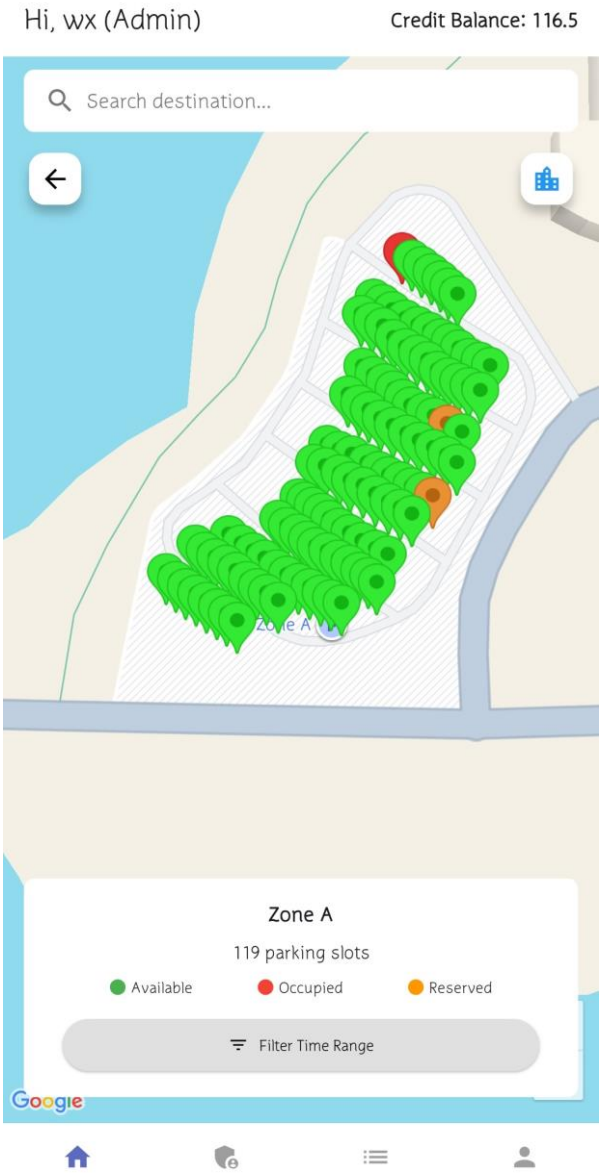


Figure 5.4.3.1 Real-Time Slot Availability Display

When the user taps on a zone marker in Figure 5.4.2, the system displays all individual parking slot markers within that zone as shown in Figure 5.4.3.1. The slot status is synchronized with the Firestore database in real time so that any updates can be immediately reflected in the user interface. Marker colors are used to represent the slot status consistently: green for available, orange for reserved, and grey for unavailable. By default, the system displays the current slot availability. However, users may also apply a time filter to preview the expected slot availability at future time intervals to support more informed reservation decisions.

5.4.4 Reservation

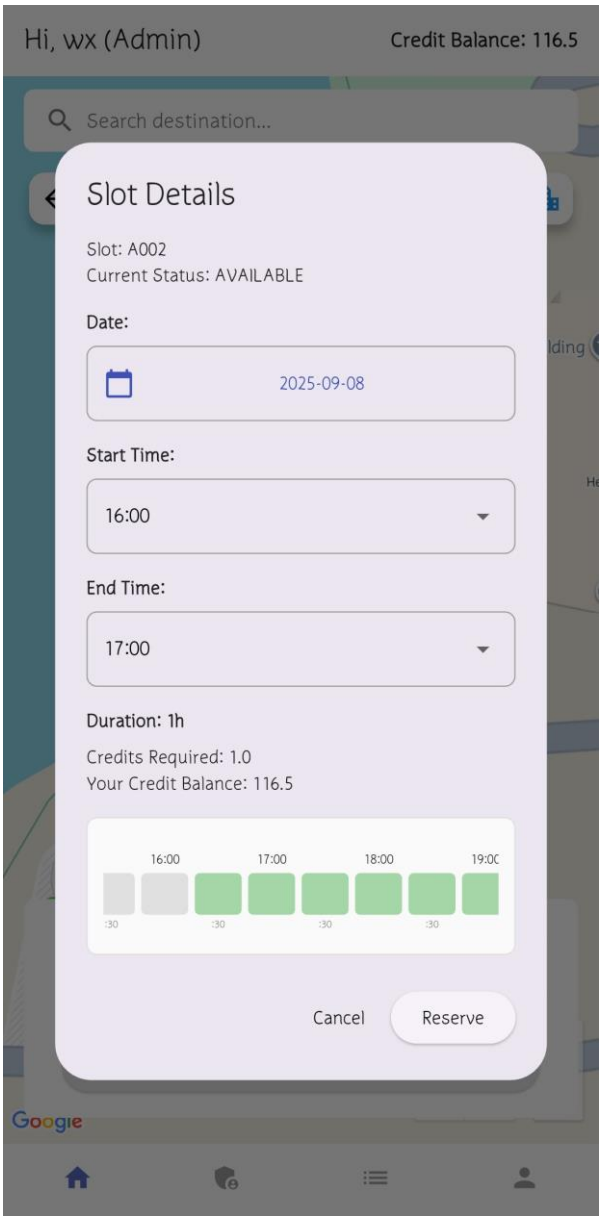


Figure 5.4.4.1 Reservation UI

Figure 5.4.4.1 illustrates the reservation dialog, which appears after the user selects a desired slot. In this dialog, the user is required to select the reservation date, start time and end time. The dialog provides key details such as the slot ID, its current status, the reservation duration, and the credits required. At the bottom, a reservation timeline is displayed to visualize the slot's availability throughout the day. This allows the user to check for existing reservations and select a suitable time. Once the details are confirmed, the user can proceed by clicking the Reserve button to finalize the booking.

5.4.5 Smart Recommendation

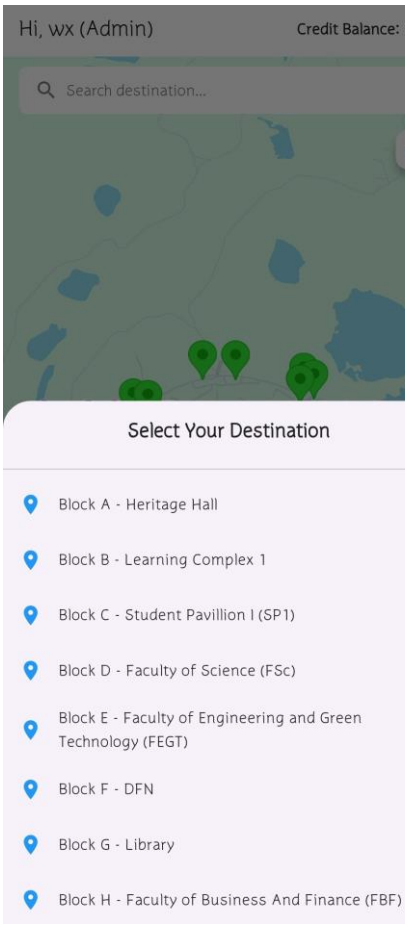


Figure 5.4.5.1 Select Destination

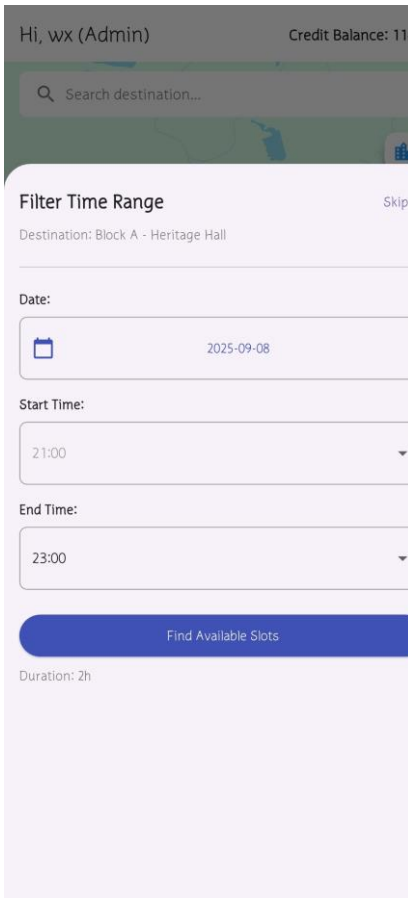


Figure 5.4.5.2 Select Time Range

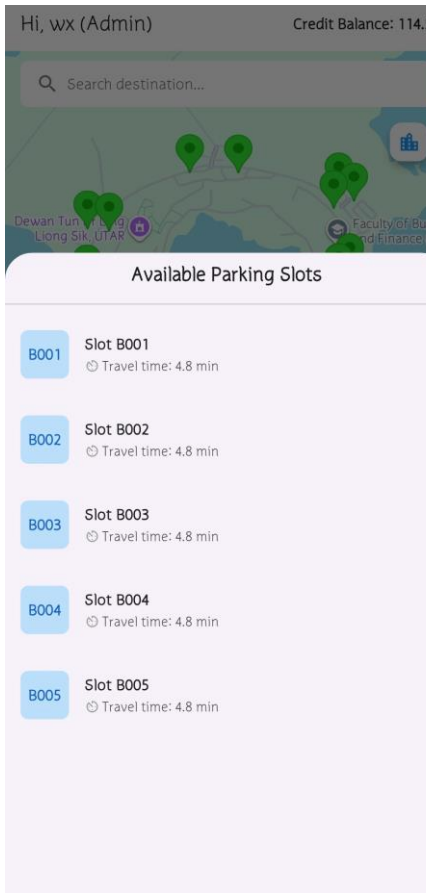


Figure 5.4.5.3 Result

Users can get system recommendations on which slot to through the home page search bar located on top of screen. They are required to select their destination building from the 15 predefined campus blocks and may optionally specify a time range if they wish to reserve a slot for a future period. The backend analyzes factors such as travel time and slot availability to generate a ranked list of suitable parking spaces. The results are displayed in a modal bottom sheet, as shown in Figure 5.4.5.3. Each recommended slot is presented with its identifier and the estimated time taken for users to reach their selected destination from their current location. Users can then tap on any recommended slot to proceed with the reservation process as the reservation dialog shown in Figure 5.4.4 will appear.

5.4.6 Reservation History Record

Figure 5.4.6.1 Reservation History

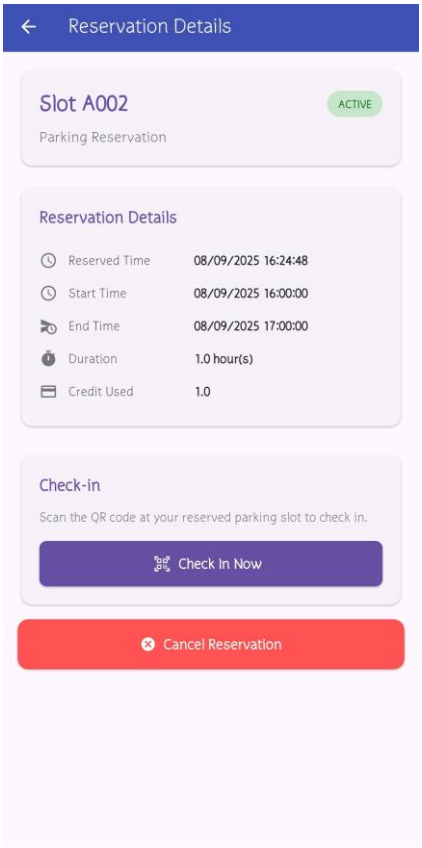


Figure 5.4.6.2 Reservation Detail

After making a reservation, users can view their records in the Reservation History section, which is divided into three tabs: Active, Completed, and Cancelled. Each record is displayed in a list format and includes details such as start time, end time, duration, and credits used. Users can tap on any record to view more detailed information. For active reservations, users are provided with several options. They can either check in or cancel their reservation if their plans change. For check-ins, the interface displays a countdown showing how many minutes remain until the user is allowed to perform the check-in. If user chooses to cancel, a refund or credit deduction will be applied according to the cancellation policy. However, cancellations are not permitted if the remaining reservation duration is less than 30 minutes. Active reservations also include a navigation button that allows users to view the route and receive directions to their reserved slot. Completed reservations are retained for reference, allowing users to review their past reservations, including slot details, reservation time, and credits spent. For cancelled reservations, users can view the records of their previously cancelled bookings along with details such as the policy type applied, the number of credits refunded, and any penalty credits imposed.

5.4.7 QR code check-in

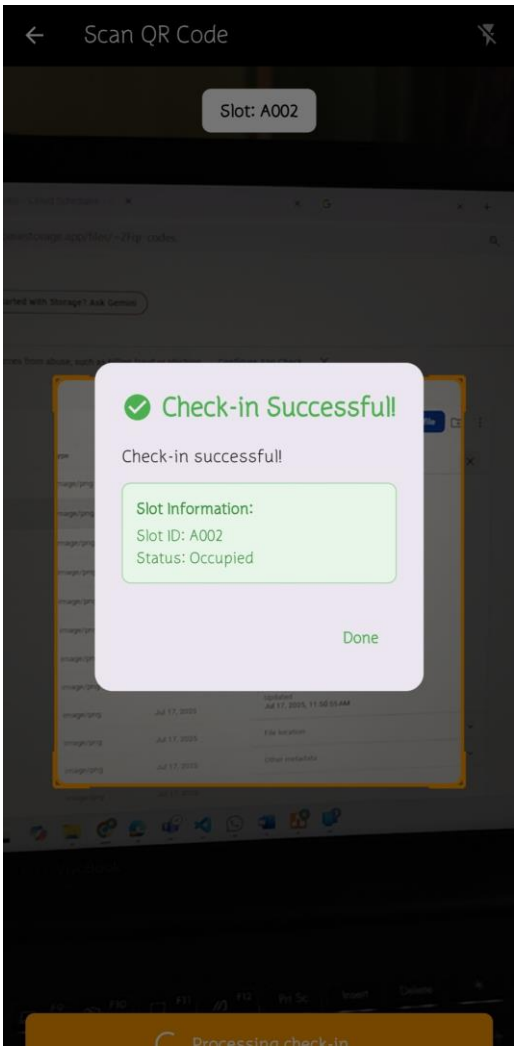


Figure 5.4.7.1 Check in Process

Figure 5.4.7.1 illustrates the check-in process for a reserved parking slot. Users are allowed to check in up to 15 minutes before their reservation starts if the slot is not being occupied. The system validates the reservation against the slot ID embedded in the QR code to ensure that the correct user checks in to the correct slot. Once validated, the reservation status and slot status are updated in Firestore and a check-in log is recorded with the exact timestamp. This mechanism is used to provide secure, real-time verification and prevents unauthorized usage of reserved slots.

5.4.8 Navigation

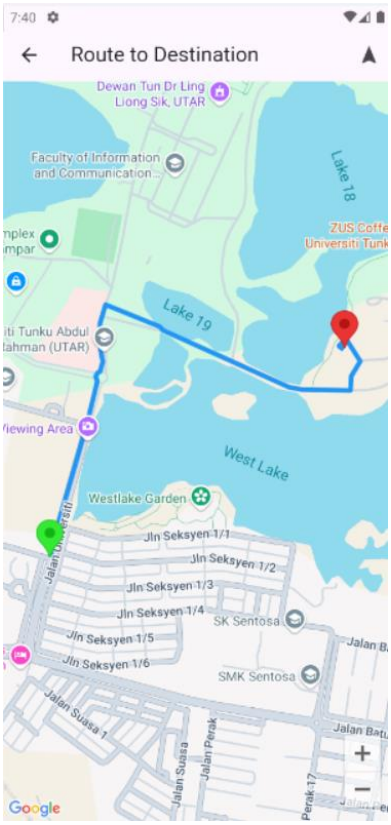


Figure 5.4.8.1 Preview Route

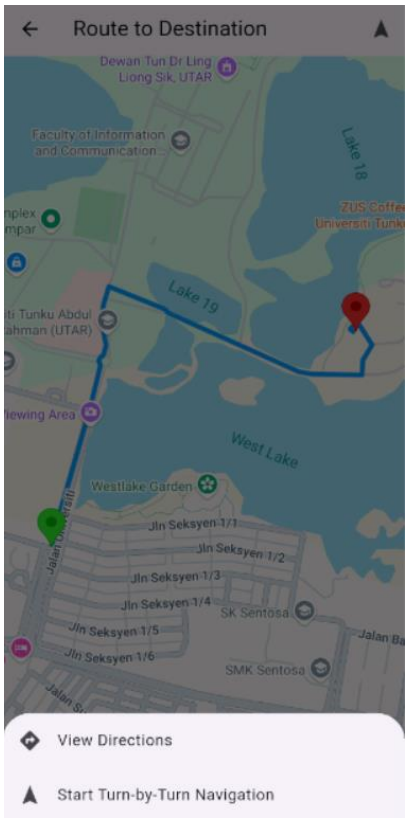


Figure 5.4.8.2 View Direction

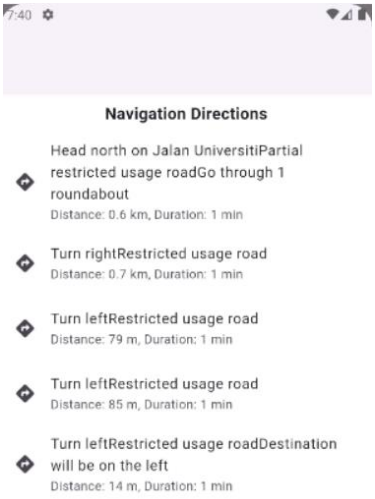


Figure 5.4.8.3 Route Detail

The navigation button is available for each active reservation within the active reservation interface. When the user clicks this button, the app requests location permission to access the user's current position. Once user grant the location permission, the system provides two navigation options: route preview or turn-by-turn navigation. If the user picks the route preview option, the system shows the route on Google Maps within the app. At the top right corner, a button enables the user to open a menu and view detailed step-by-step directions. Alternatively, if the user selects turn-by-turn navigation, the app redirects Google Maps to initiate navigation in real-time to the reserved parking slot.

5.4.9 Credit Usage History

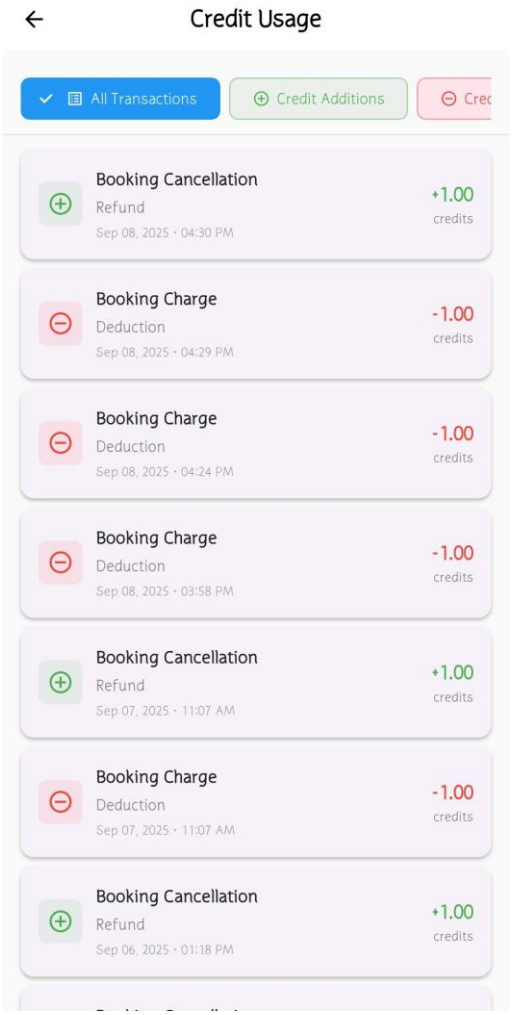


Figure 5.4.9.1 Credit Usage

In the user profile, users can access the credit usage page to view their transaction history related to parking credits as shown in Figure 5.4.9.1. This section records all credit activities which include credit additions such as refunds or resets, and credit deductions such as penalties or reservation usage. Each transaction is displayed with details including the reason, transaction type, amount and the time of the activity. Users can apply filters to view all transactions, additions, or deductions. The system supports infinite scrolling for browsing past records, as well as a refresh option to ensure the latest transactions are displayed. This feature is implemented to enable users to conveniently monitor and manage their credit balance within the application.

5.4.10 Setting

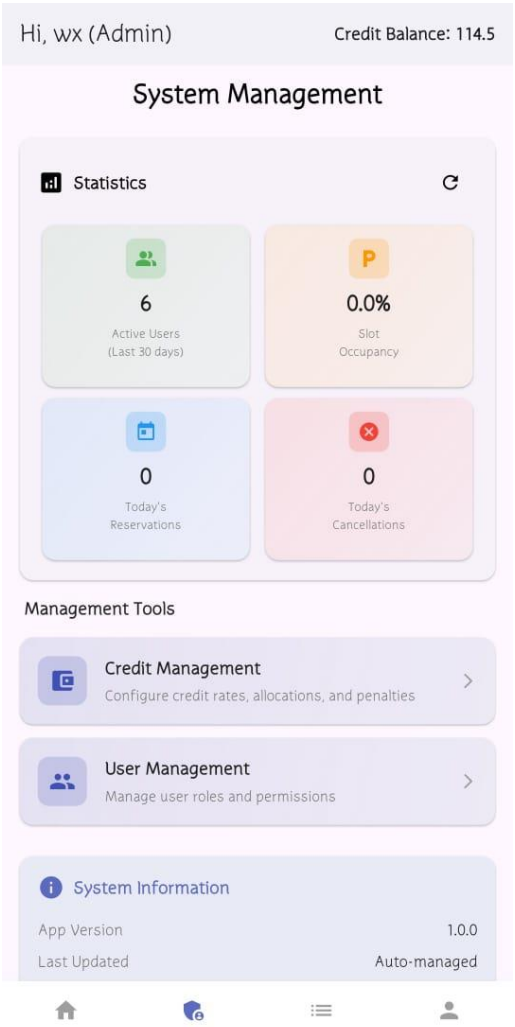


Figure 5.4.10.1 Credit Usage

Figure 5.4.10 shows the settings interface for system management, which is accessible only to administrators. This page is designed to assist the management and configuration of system features within the application. At the top, a statistics section provides administrators with a quick overview of parking resource utilization, including the number of active users, overall slot occupancy, today's reservations, and today's cancellations. These indicators help administrators monitor system performance and usage trends effectively. Below the statistics, management tools are provided for credit and user management. Through these tools, admin can configure credit rates, allocations, penalties, and manage user roles and permissions. Finally, the page also displays basic system information, including the app version, last updated status, database connection status, and admin panel availability. This provides the admin with operational control and transparency over the system's current state.

5.4.11 Credit Management

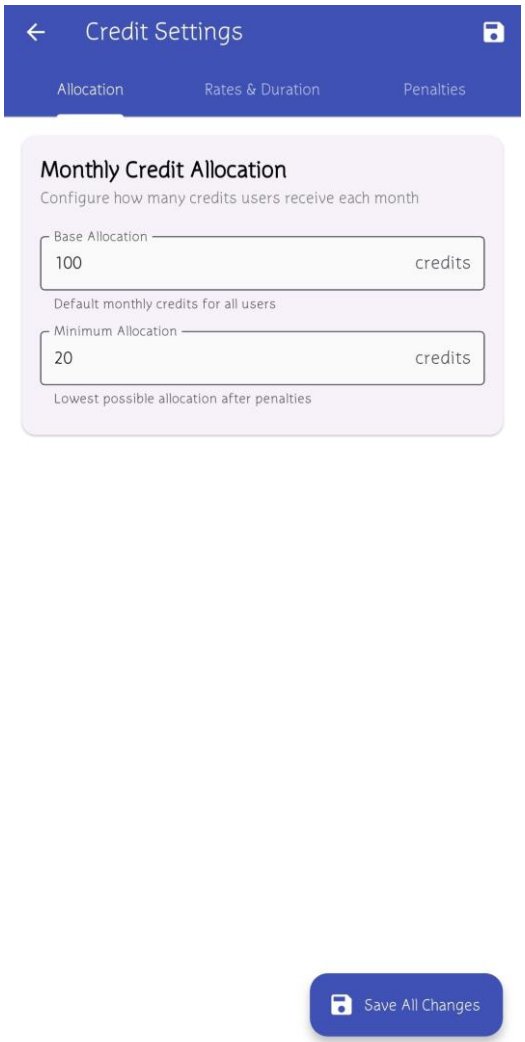


Figure 5.4.11.1 Credit Usage

The credit management subpage provides admin with access to the system's credit configuration. This section is organized into three parts: monthly credit allocation, reservation rates and duration, and penalties. Through this interface, admins can view the current configuration values and make updates when necessary. Any changes applied will be immediately reflected in the application's operation. This design make sure that the system remains flexible and adaptable to policy changes to maintain consistency across all user reservations.

5.4.12 User Management

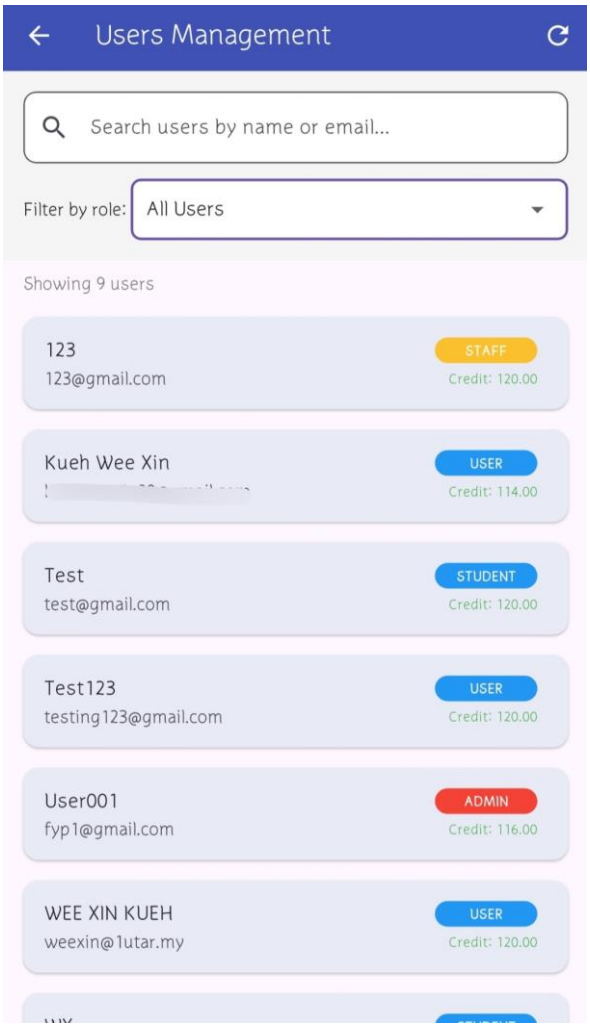


Figure 5.4.12.1 All User

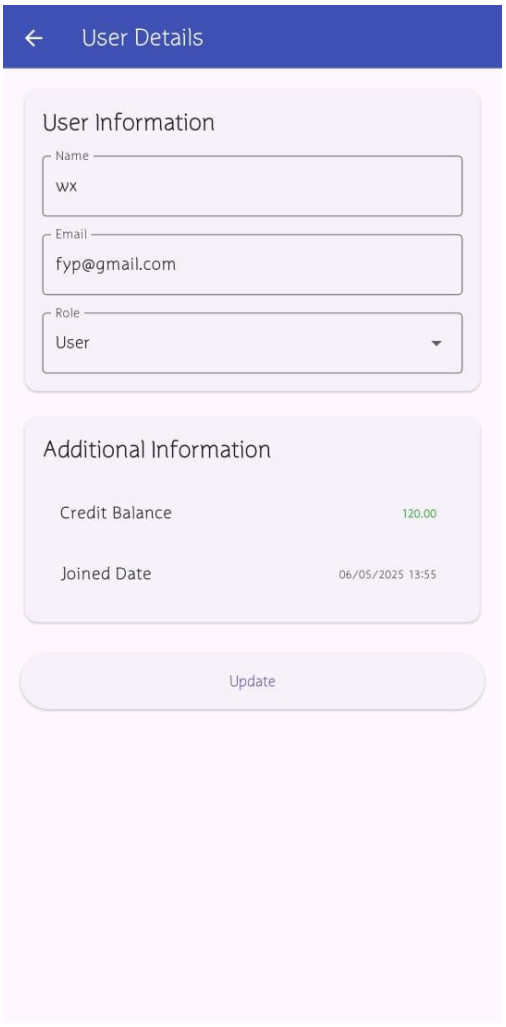


Figure 5.4.12.2 User Detail

Another management tool is the user management subpage in Figure 5.4.12. This interface displays all registered users in a list. A search bar is provided at the top to support admins to locate a specific user by name or email. Additionally, administrators can filter the list based on user roles, either user or admin. It is important for handling large datasets. When a user entry is selected, the system opens a detailed view of the user profile as presented in Figure 5.4.12.2. This page allows admins to view user information, credit balance, and date joined. Admins can also update user information such as usernames, emails, or roles. These updates are saved to the database in real time so that changes are immediately reflected across the system.

5.5 Deployed Cloud Functions

Function	Trigger	Version	Requests (24 hrs)	Min / Max Instances	Timeout
recommendSlots us-central1	HTTP Request https://recommendslots-kqt7am7j7q-uc.a.run.app	v2	0	0 / 3	1m
scheduledUpdateSlotStatus us-central1	* * * * *	v2	0	0 / 3	1m
createReservation us-central1	HTTP Request https://createreservation-kqt7am7j7q-uc.a.run.app	v2	0	0 / 3	1m
processNoShowPenalties us-central1	0 1 * * *	v2	1	0 / 3	1m
monthlyResetUserCredits us-central1	0 0 1 * *	v2	0	0 / 3	1m
cancelReservation us-central1	HTTP Request https://cancelreservation-kqt7am7j7q-uc.a.run.app	v2	0	0 / 3	1m

Figure 5.5.1 Cloud Functions

5.5.1 Scheduled Update Status Function

The scheduled update status function is deployed to maintain real-time accuracy of slot availability and reservation states. It is scheduled to run every minute. The function starts with querying all expired reservations that are still marked as active. These are reservations that should transition from active to completed status. For each expired reservation, the system checks the checkin_logs collection to determine whether the user already checked in. This step both provides data for operational analytics and triggers the no-show penalty system for users who failed to utilize their reserved slots. The function then updates the reservation status to “completed”. Slot availability is also updated based on active reservations. If a slot still has ongoing or upcoming reservations, it remains marked as reserved. If the user has checked in, the slot is shown as occupied. A slot only becomes available again when no active reservations remain. Finally, the function calculates the number of available slots for each zone and updates the corresponding data.

5.5.2 Parking Space Recommendation function

This function is used to support the smart recommendation feature in the application. Users can select their destination building on campus and receive system recommendations on which parking slot is the most suitable based on the total travel time. First, the user’s current location, destination, and optional time range are passed into the function. The function validates the destination block and retrieves its location from the database. It then calculates the estimated driving time from the user’s GPS coordinate to each parking zone using the Google Maps

Directions API. Next, the system retrieves the predefined walking time from each zone to the selected destination building. By summing up the driving time and walking time, the function calculates the total travel time for each zone. The zones are then ranked in ascending order of total time. After ranking, the function checks slot availability within each parking zone. By default, availability is verified based on the current time. If a user applies a time filter, the function checks reservation records to detect conflicts during the specified period to ensure accurate availability results. Finally, the top five suitable slots are returned, with details such as zone name, slot identifier, estimated driving and walking times, and the slot's location.

5.5.3 Create Reservation Function

The reservation process is initiated from the frontend when a user clicks the Reserve button, which triggers this backend Cloud Function. As part of the system architecture, this function ensures proper reservation management by applying business rules and preventing duplicate bookings. Upon being called, the function receives the user id, selected slot, start time, end time, and duration as input parameters. It first validates the provided data and calculates the required credits for the reservation based on the system's credit configuration. The system also ensures that the requested duration falls within the configured minimum and maximum limits. Before processing the reservation, the function performs conflict detection by querying existing active reservations for the selected slot. This validation checks whether the new reservation would interfere with existing reservations to prevent double-bookings and ensures each slot can only be reserved by one user at a time. If the input is valid and no conflicts are found, the system deducts the appropriate credits from the user's balance, creates a new booking record in the database, and updates the slot status accordingly. Upon successful completion, the function returns detailed reservation information such as reservation id, credit cost, remaining user balance, and confirmation of the booking times. The system also generates audit logs for both the reservation creation and credit transaction. This ensures that reservations are processed consistently, fairly, and securely in real time.

5.5.4 Process No Show Penalties

This function is responsible for handling cases where a user reserves a slot but fails to check in and use it. It automatically detects no-show incidents and applies penalties to the responsible users. The detection process will run daily to ensure timely enforcement. The penalty application will escalate consequences based on the user's offense history over the previous six

months. For first-time offenders, the system applies to a base penalty of 5 credits, and the user will lose their reserved credits. Repeating offenses trigger exponentially higher penalties, with a maximum cap of 50 credits to prevent excessive punishment. For users with more than three offenses, the system may additionally reduce their monthly credit allocation. All penalties detailed records are stored in noShowLogs and creditLogs collections for transparency and auditing. This approach is crucial to ensure fairness, discourages misuse, and promotes consistent adherence to reservation rules.

5.5.5 Monthly Reset User Credits

When a new user registers in the system, they are provided with an initial allocation of credits that can be used to make reservations. Monthly reset user credit's function ensures that users receive their monthly parking credit allocation at the beginning of each month. Any unused credits from the previous month are not carried forward to encourage users to plan their reservations wisely within the monthly limit. The reset process updates each user's balance according to the configured base or customized allocation and records the transaction in the credit logs for transparency. By enforcing this reset, the system maintains fairness among all users and prevents accumulation of unused credits over time.

5.5.6 Cancel Reservations

After creating a reservation, users are given the option to cancel it before they perform check-in. This function validates whether the user has already checked in for the reservation. If a check-in record exists, the cancellation request is rejected. If no check-in has been performed, the function proceeds with the cancellation process. To prevent last-minute changes, cancellations are not allowed if there are less than 30 minutes remain before the reservation ends. The function then evaluates the applicable cancellation policy to determine the amount of credits to be refunded and whether penalty credits should be applied in cases such as late or in-reservation cancellations. The refund and penalty are calculated dynamically based on factors such as the time remaining before the reservation starts, how much of the reservation has already elapsed, and any grace period rules. Once the cancellation outcome is determined, the reservation status is updated, the slot may be updated to available, and both a cancellation log and a credit log are recorded for auditing purposes.

5.6 Implementation Issues and Challenges

During the process of developing the parking reservation system, there are several implementation challenges that have been encountered and require creative problem-solving methods. The first major challenge was determining how to effectively visualize the location of each parking slot and zone across the campus for mobile users. Initially, the development process involved attempts to create detailed floor plans and overlay interactive buttons on campus maps to provide users with precise visual representations of parking areas. However, this approach quickly proved problematic as drawing detailed floor plans would have required substantial design time. This will ultimately be diverting focus from the core reservation and credit system functionality that formed the main objectives of the project.

When attempting to implement the overlay button approach in Flutter, several critical usability issues emerged. The mobile screen size limitations meant that users would need to zoom in and out frequently to navigate the large campus map effectively. However, these zooming actions caused the button overlays to become inaccurate and misaligned with their intended positions which resulted in a poor user experience as users might accidentally select wrong parking slots or be unable to interact with the interface reliably. As an alternative solution, experimentation with a grid view approach displayed parking slots as a series of buttons that allowed users to tap directly on slots to proceed with reservations. However, this method created its own set of usability problems. Users could only see slot id without understanding the actual physical locations of their chosen spots until after completing their reservations and using the navigation feature.

The final solution implemented a hierarchical location display system using google maps integration that effectively addressed all the previous challenges. The home screen now displays a clean map interface with 17 strategically placed parking area icons representing different zones across the campus. When users tap on specific zones, the system reveals individual slot locations within that area. This approach successfully manages the display of over 2000 parking spaces without overwhelming the user interface. Meanwhile, real-time slot status is clearly communicated using different colored location icons to indicate availability, occupied, or reserved status. This solution enables users to make informed reservation decisions with clear visual context of slot locations and their relationship to campus landmarks.

The second significant implementation challenge is the management of system configurations and policies. Initially, all system settings were hard coded directly into the application code. These include critical values such as reservation rates and duration, monthly credit allocations, cancellation policies, and penalty structures. This approach created maintenance challenges as any policy updates required code modifications followed by complete application redeployment. The hard-coded configuration approach also lacked the flexibility needed for testing different pricing strategies or making real-time adjustments based on evolving usage patterns and administrative requirements.

To resolve these configuration management issues, the development process involved creating a comprehensive dynamic configuration management system that migrated all policy values and system settings to the database. The backend code was restructured to reference database values for all policy calculations and business logic operations to eliminate the dependency on hard-coded values. Additionally, the implementation included an admin interface that allows authorized administrators to update system configurations through an intuitive user interface without requiring any technical knowledge or development intervention. The system also incorporates configuration caching mechanisms to ensure optimal performance while maintaining the flexibility of database-driven configurations. This solution provides significant operational flexibility, reduces deployment requirements for routine policy adjustments, and enables rapid response to changing business requirements or user feedback regarding system policies.

5.7 Concluding Remark

Chapter 5 shows how the system design outlined in the previous chapter was implemented to build the smart parking solution. The development process integrated multiple technologies including Flutter for cross-platform mobile development, Firebase services for backend functionality, and Google Maps for location-based features. The data collection phase formed the foundation of the system, where precise GPS coordinates for over 2,000 parking slots across 17 zones were collected and mapped with campus infrastructure. The Firebase backend configuration provides scalable cloud-based services, including real-time database synchronization, user authentication, and serverless cloud functions that handle complex and critical business logic such as reservation processing and credit system operations. The deployment of six cloud functions ensures reliable backend operations for scheduled status updates, parking recommendations, reservation processing, penalty management, credit resets, and cancellation handling. For the mobile app, the user interface seamlessly integrates reservation, smart recommendations, real-time navigation, check-in processes, and comprehensive credit and user management features.

Despite challenges in visualization complexity and configuration management, feasible solutions were developed that maintained full system functionality. The hierarchical location display system effectively manages the complexity of displaying thousands of parking slots and the dynamic configuration management system provides admin flexibility without requiring redeployment or code modifications. In addition, the QR code check-in system offers a practical alternative to costly IoT infrastructure to maintain the system integrity and create a foundation for future hardware integration. This approach demonstrates that effective parking management solutions can be implemented cost-effectively while remaining scalable for future technological improvement. In the next chapter, comprehensive testing and evaluation are presented to validate the effectiveness of implemented system.

Chapter 6

System Evaluation and Discussion

6.1 System Testing and Performance Metrics

System testing and performance metrics are conducted to ensure that the parking reservation and management system works correctly and meets user expectations. The testing in this project focuses on frontend performance and database performance. Performance metrics are also used to measure the efficiency and responsiveness of the mobile application and the backend cloud services. The key aspects evaluated include app start time, screen rendering, Firestore query and transaction speed, and cloud function latency, memory usage and CPU usage. These evaluations help identify the system stability and highlight areas for improvement.

6.1.1 Frontend Performance Analysis

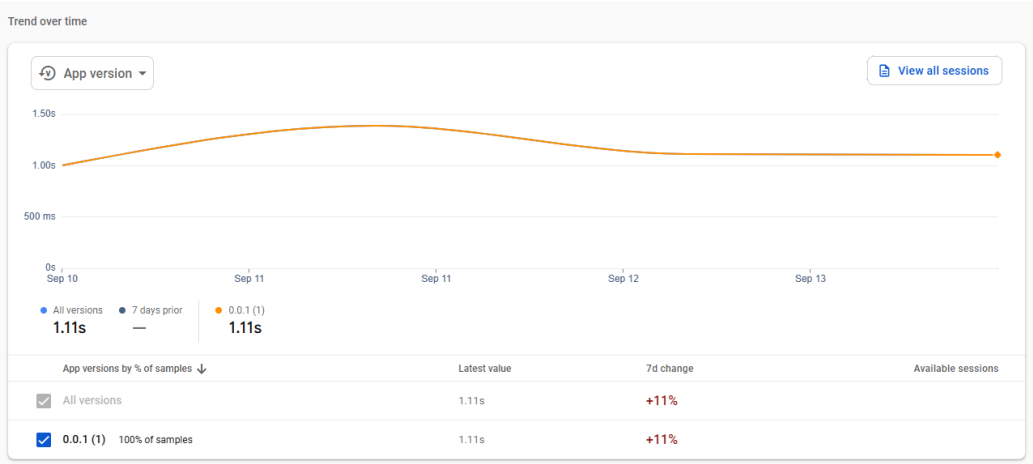


Figure 6.1.1 App Start Time

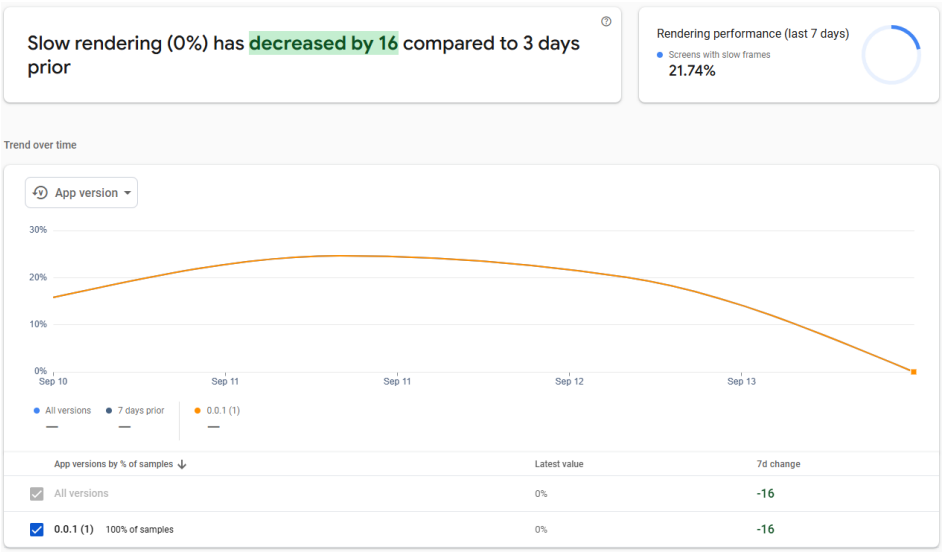


Figure 6.1.2 Slow Rendering

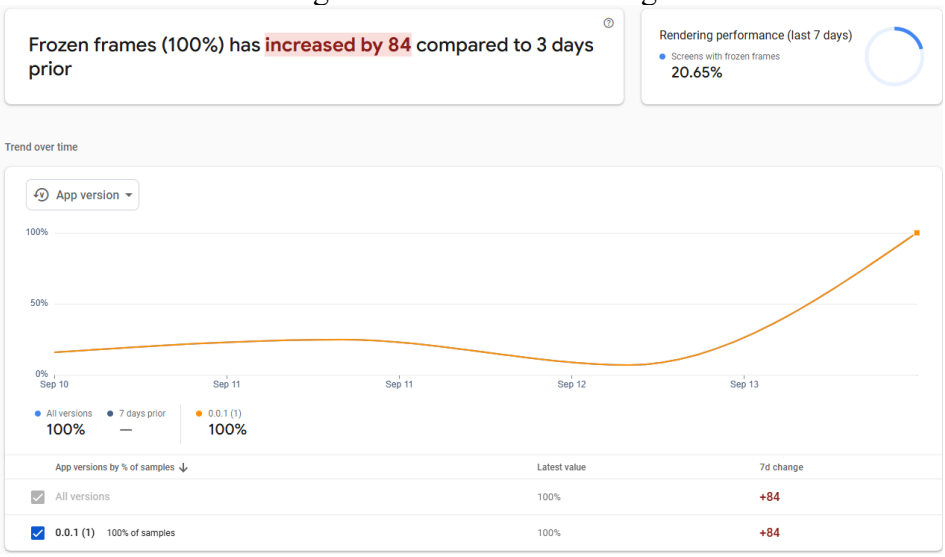


Figure 6.1.3 Frozen Frame

Frontend system performance was evaluated based on app start time and screen rendering quality. The average app start time recorded over the past seven days was 1.11 seconds. It is within an acceptable range and indicates that the application launches quickly from the user's perspective. Screen rendering performance was assessed using two metrics: slow frames and frozen frames. The results show that 21.74% of screens experienced slow frames which means that more than one frame in five frames took longer than 16ms to render. In addition, 20.65% of screens contained frozen frames, where at least one frame took longer than 700ms to render. These percentages suggest that the app launches quickly but there are performance concerns in rendering responsiveness as both slow and frozen frames exceed the ideal threshold for a smooth user experience.

6.1.2 Database Performance Testing

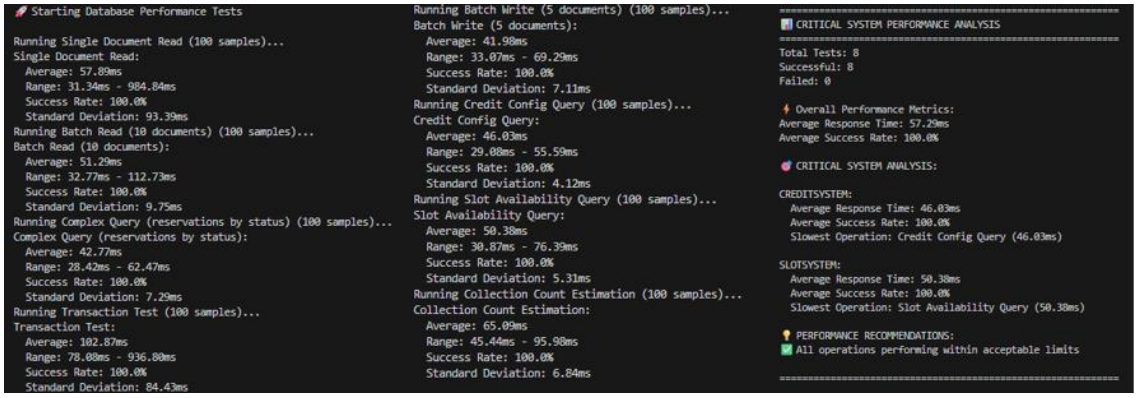


Figure 6.2.1 Database Performance Testing

The database performance testing was conducted across a series of Firestore operations. The testing includes single document reads, batch reads, complex queries, transactions, batch writes, configuration queries, slot availability queries, and collection count estimations. The results indicate that the overall database performance is stable and reliable as the average response time is 57.29ms and a 100% success rate across all tests. For single document reads, the average response time was 57.89ms, but the maximum recorded latency achieved 984.84ms. This suggests that while most reads are efficient, there can be outliers likely due to network conditions or backend load. Batch reads and complex queries performed well, the average is 51.29ms and 42.77ms respectively. The low standard deviation indicates consistent and predictable response times.

Transactions had the highest average response time at 102.87ms and the range of variability is 78.08ms to 936.80ms. This is expected since transactions involve multiple reads and writes that make them slower and more variable. In contrast, batch writing was efficient, averaging only 41.98ms with minimal variance which means that write operations are highly optimized.

For domain-specific operations, the credit system query averaged 46.03ms and the slot availability query averaged 50.38ms. The collection count estimation was slightly higher at 65.09ms. Overall, the database performance testing confirms that Firestore operations in this project are efficient, stable, and within acceptable performance limits. The only concern lies in occasional outliers observed in single document reads and transactions, but these do not significantly affect average performance. The results demonstrate that the system's database layer can support real-time interactions, such as reservation handling and credit management, without noticeable delays to the end users.

6.1.3 Performance Metrics

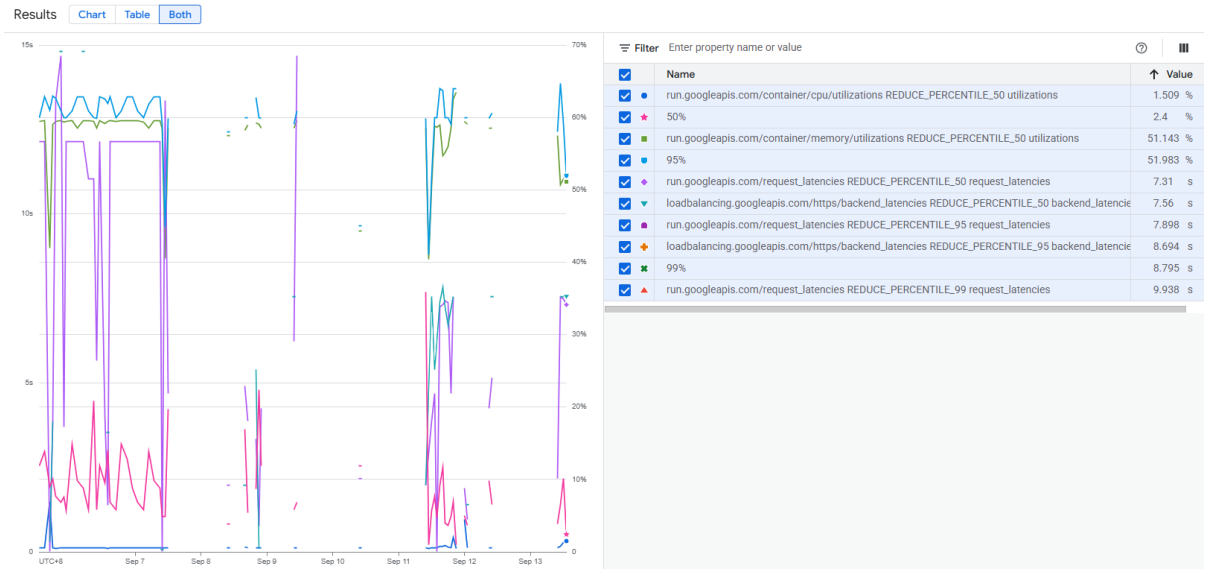


Figure 6.3.1 Performance metrics chart

Table 6.3.1 Performance Metrics

Cloud Function	P50	P95	P99
Request Latency (s)	7.31	7.898	9.938
Backend Latency (s)	7.56	8.694	8.795
Memory Utilization (%)	51.143	-	51.983
CPU Utilization (%)	1.509	-	2.4

Figure 6.3.1 illustrates the performance chart, while the detailed data is summarized in Table 6.3.1. The performance metrics, collected over a one-week period from 5 September to 13 September, include request latency, backend latency, memory utilization, and CPU utilization of the cloud function.

First, request latency refers to the total time for a client's request to obtaine a complete response from the cloud function, including both network transfer and execution time. For the cloud function deployed in this project, the recorded values are 7.31s (p50), 7.89s (p95), and 9.93s (p99). Most of the requests are completed in around 7 seconds, but about 1% of requests take up to 10 seconds to finish. Although the latency is relatively consistent, the response time is still considered slow for real-time applications.

Besides that, backend latency measures the time spent executing the function logic itself, excluding network overhead. The results are 7.56s (p50), 8.69s (p95), and 8.79s (p99). These values are very close to the request latency, which means that most of the delay comes from function execution rather than network transfer. This suggests that performance improvements should focus on optimizing function logic instead of network configurations.

Additionally, memory utilization indicates the percentage of allocated memory consumed during execution. The recorded values are 51.14% (p50) and 51.98% (p99), showing stable and moderate usage. Since the utilization is well below 100%, the allocated memory is sufficient, and there is no immediate risk of memory-related performance issues.

Lastly, CPU utilization represents the proportion of CPU resources used by the function. The values are low, with 1.51% (p50) and 2.40% (p99). This shows that CPU resources are underutilized and not a limiting factor in performance. The results confirm that the high latency observed is caused by the execution logic of the cloud functions themselves.

6.2 Testing Setup and Result

After evaluating the system performance, functional testing was conducted to verify that all features of the parking reservation and management system operated as intended. The testing focuses on how users interact with the application. Each test case was designed with clear inputs, the expected outputs, and the results to verify the accuracy of the system.

6.2.1 User Log in / Sign in

Table 6.2.1 User Authentication Test Case

Test case	Description	Input	Expected Output	Result
1	New user sign in	Click "Sign in with Google" and complete sign-in	1. User signed in 2. New user doc created in database with 100 credits 3. Navigates to Home Page	PASS
2	Existing user sign-in	Click "Sign in with Google" and complete sign-in	1. User signed in 2. User data corrects 3. Navigates to Home Page	PASS
3	User cancels sign-in	Click "Sign in with Google" and cancel	1. No sign-in 2. Stays on LoginPage	PASS
4	Sign-in error	Click "Sign in with Google" with invalid setup	1. Error message shown 2. Stays on LoginPage	PASS

The user authentication test cases were designed to verify that the sign-in process works properly for both new and existing users. The tests covered normal scenarios such as successful sign-in and exception cases such as sign-in cancellation and invalid configuration errors. Based on the results, the system successfully created a new user profile with the correct initial credits, loaded existing user data accurately, and handled cancellation and error conditions as expected.

6.2.2 Reservation

Table 6.2.2 Reservation Test Case

Test case	Description	Input	Expected Output	Result
1	Load zones on map	Open Reservation Page	1. Zone markers appear on maps 2. Each shows name and available slots	PASS
2	View slots in a zone	1. Tap a zone marker 2. click "View Slots"	1. Map zooms to zone 2. Slot markers appear 3. Zone info card shows slot count	PASS
3	Apply time filter	1. Select zone 2. click "Filter Time Range" 3. choose valid date and time range 4. click "Apply"	1. Slot markers update color to show filtered statuses. 2. Filter details shown in zone info card	PASS
4	Create reservation (sufficient credits)	1. Select available slot 2. choose valid date/time 3. click "Reserve"	1. Reservation created 2. Success message display	PASS
5	Create reservation (insufficient credits)	1. Select available slot 2. choose valid date/time with credits exceeding user balance 3. click "Reserve"	1. Display error message 2. No reservation created 3. Slot status unchanged	PASS

The reservation feature was tested to ensure that users can correctly view all parking areas on campus, check parking space availability in real time and make reservations under different conditions. After login, users are directed to the home page, where the integrated google map successfully displays all zones with location markers indicating the zone names and the number of available slots. When a zone marker is selected, the map zooms in to the correct area and the zone details are shown. The time filter function also works as expected as users can choose a valid time range and apply it to check the slot status for the selected period. For booking a parking space, the system correctly creates a reservation and displays a successful message when the user has sufficient credit, and the selected slot is not already reserved by another user. However, in cases of insufficient credit, the reservation process fails, and an error message is displayed.

6.2.3 Smart Recommendation

Table 6.2.3 Smart Recommendation Test Case

Test case	Description	Input	Expected Output	Result
1	Recommendation without time filter	1. Enable location access 2. Tap "Search destination" 3. Select Block A from list 4. Tap "Skip" on time filter	1. Shows loading dialog 2. Returns list of available parking slots 3. Slots sorted by total travel time 4. Shows total travel time for each slot	PASS
2	Recommendation with time filter	1. Enable location access 2. Select Block B as destination 3. Set date to tomorrow 4. Set start time to 08:00 5. Set end time to 10:00 6. Tap "Find Available Slots"	1. Shows slots available for specified time 2. Only displays slots free from 08:00-10:00 3. Sorted by total travel time 4. Shows travel duration	PASS
3	Reservation	Tap one of the recommended slots.	1. Reservation dialog is opened. 2. Users can proceed with the reservation process.	PASS
4	No available slots	1. Select destination with no available slots 2. Complete search process	1. Shows message "No parking recommendations available" 2. Empty recommendations list.	PASS
5	Location permission denied	1. Deny location permission 2. Tap "Search destination"	1. Shows error message "Location permission is required" 2. No destination picker shown 3. Prompts user to enable location	PASS

The functional testing then proceeded to the smart recommendation feature. This feature is accessed through the reservation page via the search bar located at the top of the home page with integrated google map. To use this feature, user's current location is required. If location permission is denied, the system correctly displays an error message and prompts the user to enable it before continuing. To receive system recommendations, users must select their destination building on campus from a predefined list of options and optionally apply a time filter. Based on the tests, users were able to successfully receive 5 suitable recommendations for currently available parking slots which were sorted by the shortest total travel time when no time filter was applied. When a time filter was applied, the system correctly returned recommendations for slots that were available during the specified future time range. Users could also proceed with the reservation process directly from the recommendation results. In cases where no slots were available, the system displayed an appropriate error message indicating that no parking recommendations were found.

6.2.4 Reservation History

Table 6.2.4 Reservation History Test Case

Test case	Description	Input	Expected Output	Result
1	View Active Reservations	1. Navigate to "My Reservations" 2. Tap "Active" tab	1. Shows all active reservations 2. Displays slot ID, start/end time, duration, credits 3. Shows navigation button for each reservation 4. Shows swipe-to-cancel functionality	PASS
2	View Completed Reservations	1. Navigate to "My Reservations" 2. Tap "Completed" tab	1. Shows all completed reservations 2. No navigation or cancel buttons 3. Displays check-in information if available 4. Sorted by creation date (newest first)	PASS
3	View Cancelled Reservations	1. Navigate to "My Reservations" 2. Tap "Cancelled" tab	1. Shows all cancelled reservations 2. No action buttons available 3. Displays cancellation details when tapped 4. Shows refund/penalty information	PASS
4	View Reservation Details	Tap on any reservation from any tab	1. Opens detailed view 2. Shows all reservation information 3. Displays status badge with correct color 4. Shows check-in status if applicable	PASS
5	No Show Details	Open completed reservation details	1. Shows offense count 2. Shows penalty credits deducted 3. Shows penalty applied timestamp	PASS
6	Cancellation Details	Open cancelled reservation details	1. Shows cancellation policy applied 2. Displays refund percentage and amount. 3. Shows penalty credits deducted 4. Shows cancellation timestamp	PASS

After successfully making a reservation, users can view their reservation history on the “my reservations” page. The page is divided into three tabs: active, completed and cancelled. In the active tab the system displayed all ongoing reservations with details such as slot id, start and end times, duration and credits used. It also provided navigation and swipe-to-cancel options. The completed tab showed all finished reservations sorted by creation date. If a user had checked in, the check-in details were displayed. For completed reservations cancel or navigation options are not provided as expected. The cancelled tab listed all cancelled reservations where users could view cancellation details, refund amounts and penalty information if applicable. For every reservation, tapping on the record opened a detailed view showing reservation data, status indicators and check-in status where relevant. When no-show penalties or cancellation policies were applied, the details were shown clearly with offense counts, deducted credits, timestamps and the applied policy.

6.2.5 Cancel reservation

Table 6.2.5 Cancel Reservation Test Case

Test case	Description	Input	Expected Output	Result
1	Cancel Reservation (Swipe)	1. Go to Active Tab 2. Swipe left on an active reservation	1. Shows red cancel background 2. Displays confirmation dialog 3. Successfully cancels if confirmed 4. Shows loading overlay during cancellation	PASS
2	Cancel Reservation (Button)	1. Open active reservation detail. 2. Tap the cancel reservation button	1. Displays confirmation dialog 2. Successfully cancels if confirmed 3. Shows loading overlay during cancellation	PASS
3	Cancel Confirmation Dialog	1. Attempt to cancel active reservation 2. Choose "No" in confirmation dialog	1. Cancellation process stops 2. Reservation remains active 3. No credits are deducted	PASS

There are two cancellation methods provided. The first is in the active tab, where swiping left on a reservation correctly displays the red cancel background and opens a confirmation dialog. When the user confirms, the reservation is cancelled, and a loading overlay is shown during the process. The second method is through the reservation detail page, where tapping the cancel button triggers the same flow. In both cases, the system consistently shows a confirmation dialog and handles the cancellation as expected. For exception handling, the confirmation dialog was also tested by selecting the “no” button, which correctly stopped the cancellation process, kept the reservation active, and ensured no credits were deducted.

6.2.6 Check in Parking Space

Table 6.2.6 Cancel Reservation Test Case

Test case	Description	Input	Expected Output	Result
1	Check-in Availability	1. Open active reservation details. 2. View check-in section	1. Shows "Check In Now" if within allowed time 2. Shows countdown if too early 3. Shows "Already Checked In" if completed. 4. Shows "Reservation Expired" if past end time	PASS
2	QR Code Check-in	1. Open active reservation with valid check-in time 2. Tap "Check In Now"	1. Open QR scanner 2. Validates reservation and slot match	PASS

		3. Scan QR code at slot	3. Records check-in time 4. Updates UI to show checked-in status	
3	Successful QR check-in	1. Open active reservation 2. Tap "Check In Now" 3. Scan valid QR code for reserved slot. 4. QR contains correct slot ID and type	1. Shows "Check-in Successful!" dialog. 2. Updates reservation status to check in. 3. Sets slot status to "occupied". 4. Records check-in timestamp	PASS
4	Invalid QR code format	1. Open QR scanner 2. Scan QR code with invalid JSON format	1. Shows "Invalid QR code format" error 2. Allows users to try again 3. Scanner remains active	PASS
5	Wrong slot QR code	1. Open QR scanner for slot A001 2. Scan QR code for slot B001	1. Shows error "No valid reservation found for slot B" 2. Provides option to try again 3. Does not perform check-in	PASS
6	Slot already occupied	1. Try to check-in to slot already occupied by another user. 2. Scan QR code	1. Shows "Slot is currently occupied by another user" 2. Check-in fails 3. Option to try again provided	PASS
7	QR scanner camera permissions	1. Deny camera permission 2. Try to open QR scanner	1. Shows permission request dialog 2. Cannot scan without camera access 3. Provides instructions to enable permission	PASS
8	Navigation after successful check-in	1. Complete successful check-in 2. Tap "Done" in success dialog	1. Closes scanner page 2. Returns to reservation details 3. Updates check-in status immediately	PASS

Table 6.2.6 illustrates the test cases conducted for the check-in process. The system correctly displayed the check-in options based on the reservation status and timing. This includes showing check in button within the allowed period, a countdown when it was too early, and status messages for completed or expired reservations. The qr code check-in feature was verified by scanning a valid code which successfully updated the reservation and slot status while recording the timestamp. Error handling was also tested such as invalid qr code formats, scanning the wrong slot code, and attempting to check in to an already occupied slot. In each case, the system displayed the appropriate error messages and provided options to retry without completing the check-in. For the case that denying camera access, a request dialog with clear instructions will be triggered for enabling permissions. After a successful check-in, the navigation flow worked as expected, returning the user to the reservation details page with the updated status.

6.2.7 Navigation

Table 6.2.7 Navigation Test Case

Test case	Description	Input	Expected Output	Result
1	Navigate to Parking Slot	1. Go to active tab 2. Tap navigation button on reservation	1. Opens navigation to parking slot 2. Shows error if navigation fails 3. Button only appears for active reservations	PASS
2	Successful navigation to parking slot	1. Tap navigation button on active reservation 2. Grant location permission when prompted 3. Select navigation option	1. Shows navigation options modal 2. Displays "View Route Preview" and "Start Turn-by-Turn Navigation" 3. Successfully gets user location and slot coordinates	PASS
3	View route preview	Select "View Route Preview"	1. Shows visual route on map 2. Displays estimated travel time and distance	PASS
4	Start turn-by-turn navigation	Select "Start Turn-by-Turn Navigation"	1. Opens external Google Maps app 2. Launches with driving directions pre-loaded 3. User can follow turn-by-turn instructions	PASS
5	Google Maps app not installed	1. Device without google maps app 2. Select "Start Turn-by-Turn Navigation"	1. Shows error "Could not launch navigation" 2. Falls back gracefully	PASS

The test cases conducted for the navigation feature are listed in table 6.2.7. The system successfully displayed the navigation button only for active reservations and correctly opened the navigation flow when tapped. Users were prompted to grant location permissions as expected. The system displayed a modal with two options for in-app route preview and turn by turn navigation. The route preview function showed a visual map route along with the estimated travel time and distance while the turn-by-turn option launched google maps with the slot coordinates preloaded for driving directions. If google maps were not installed on the device, the system displayed an alert message and gracefully fell back to the route preview option to ensure that the users could still access navigation guidance.

6.2.8 System Setting

Table 6.2.8 System Setting Test Case

Test case	Description	Input	Expected Output	Result
1	Admin access verification	1. Sign in as admin user 2. Navigate to settings page	1. Displays full settings interface with statistics 2. Shows management tools section 3. Load system information	PASS
2	Statistics display and refresh	1. Access settings as admin 2. View statistics section 3. Tap refresh button	1. Shows 4 stat cards with current data 2. Displays of active users, slot occupancy, today's reservations, cancellations 3. Refresh updates all statistics	PASS
3	Credit Management access	1. Go to settings page as admin 2. Tap "Credit Management" card	1. Navigates to Credit Setting Page 2. Shows credit configuration options 3. Allows modification credit configuration 4. Proper navigation back to settings	PASS
4	User Management access	1. Access settings as admin 2. Tap "User Management" card	1. Opens user management page 2. Shows all users in a list 3. Can search users by using the search bar 4. Tap the user card to see the user details and can make modification	PASS
5	System information display	Scroll to bottom of settings page	1. Shows system information card 2. Displays app version 3. Shows database status 4. Show Admin panel status	PASS

The testing process continued on the system setting testing. The feature includes the important settings for the credit system and for user management. It is accessible only to admin users. Upon signing in as an admin, the system correctly displayed the full settings interface which included the statistics, management tools, and system information. The statistics section provided data for active users, slot occupancy, today’s reservations, and cancellations. The refresh button worked properly to update changes in the statistics data. The credit management card correctly led to the credit settings page. The configuration settings were displayed correctly during testing and could be modified, and navigation back to the settings page worked as expected. Similarly, the user management card opened the user management page. It displayed all users in a searchable list and allowed viewing and updating individual user details. Finally, the system information card displayed the app version, database status, and admin panel status.

6.2.9 User Profile

Table 6.2.9 User Profile Test Case

Test case	Description	Input	Expected Output	Result
1	Load Profile Successfully	1. Logged in 2. Open profile page	1. Shows username, email, credit balance 2. Profile loads without errors	PASS
2	Upload Profile Image	1. Taps on profile avatar 2. Selects camera or gallery 3. Picks an image	1. Image uploads successfully 2. New profile picture displays 3. Shows success message	PASS
3	Access Credit Usage	Taps "Credit Usage History" card	1. Opens credit usage page 2. Shows list of credit transactions	PASS
4	View Credit Transactions	Opens credit usage page	1. Displays all credit transactions 2. Shows transaction date, amount, type 3. Shows newest transactions first	PASS
5	Apply Transaction Filter	1. Opens credit usage page 2. Taps filter chips (All/Additions/Deductions)	1. Shows only filtered transactions 2. Updates transaction count 3. Can clear filter to show all	PASS
6	Logout	Taps "Log Out" button	1. User gets logged out 2. Returns to login page 3. Session ends completely	PASS

The last testing is on the user profile interface. The profile page loaded successfully after logging in. The username, email, and credit balance are shown without errors. The profile image upload feature worked correctly, users can choose from the camera or gallery and upload, the new profile picture displayed successfully with a confirmation message. The credit usage history card correctly opened the credit usage page. All transactions are displayed with details such as date, amount, and type. The filter chips for “all,” “additions,” and “deductions” functioned as expected by showing only the filtered transactions, updating the count, and allowing users to clear the filter to return to the full list. Finally, the logout button successfully ends the user session and returns to the login page.

6.3 Project Challenge

Several issues were identified in this project. One of the key challenges is ensuring real-time parking slot availability detection. To ensure accurate data, IoT infrastructure such as infrared sensors and microcontrollers would need to be installed at each parking slot to detect occupancy and transmit data to the database in real time. However, the installation of such hardware introduces higher costs and increases system complexity. The accuracy of detection may also be influenced by environmental factors such as poor weather or physical obstructions. Due to these limitations, this component is excluded from the project scope. Instead, a simulated dataset is used to reflect the occupancy status of parking slots. While this approach simplifies implementation, it lacks the responsiveness of a real-time system. Therefore, the system assumes that all users must reserve a parking slot before parking. This allows full control of data and ensures the system can function effectively without installing IoT infrastructure, but it also limits the system's applicability in real-world environments.

Another challenge is ensuring that each parking slot can only be used by the individual who reserved it. To guarantee this, physical barriers need to be installed at each slot, and users must unlock the barrier by scanning a QR code through the app. While this approach effectively prevents unauthorized use of reserved parking spaces, the installation of these barriers leads to higher implementation costs. Hence, this approach is excluded in this project too. Ultimately, these challenges highlight issues related to the limitations of hardware integration due to cost constraints.

6.4 Objectives Evaluation

Objective 1: Reduce traffic congestion

The system successfully achieved this objective through multiple features. The real-time parking slot status display allows users to view whether spaces are empty, reserved, or occupied before departing to their destination, significantly reducing search time. However, simply displaying slot status could potentially cause multiple users to target the same empty slot, creating new congestion issues. To address this limitation, the reservation feature enables users to secure a parking space before they reach their destination, eliminating uncertainty and the need to "try their luck." The QR code check-in system ensures real-time updates when users arrive at their reserved slots, immediately reflecting the occupancy change to other users in the app. Through the combination of real-time availability checking and advanced reservation capabilities, users can plan their parking before departure and avoid driving to full parking areas. This eliminates the common practice of wandering through parking lots hoping to find empty spaces, directly addressing the primary cause of parking-related traffic congestion.

Objective 2: Improve parking resource management through a credit-based system

The second objective is successfully achieved by implementing a comprehensive credit-based management framework that ensures fair and efficient allocation of parking resources. The credit system is fully integrated into the reservation process to replace the traditional payment methods with a credit currency that creates a controlled environment where users must strategically plan their parking needs within allocated limits. The system implements a fixed monthly credit allocation for each user to prevent credit purchases to ensure equitable access and requiring users to plan their parking usage wisely throughout the month. The non-transferable nature of credits, where they expire monthly and cannot be carried forward, encourages efficient utilization and prevents hoarding and ensures consistent resource availability for all users. To prevent misuse, the system establishes clear rules including restrictions on advance bookings limited to the current month only and monthly allocation limits. A comprehensive penalty system addresses resource management through two key mechanisms: a cancellation policy that provides partial credit refunds to encourage users to release unused reservations to make spaces available for others, and no-show penalties that deduct credits from users who reserve but fail to use their slots. For repeat offenders, the system

implements escalating penalties including potential reductions in monthly credit allocations to ensure accountability and responsible resource usage. The system further enhances resource management through an admin interface that provides comprehensive oversight and control capabilities. Admin can access detailed statistics for reservations and slot occupancy. The admin panel also allows for dynamic adjustment of credit system parameters, including monthly credit allocation values, reservation rates determining credit cost per hour, and modification of cancellation policy and no-show penalty settings. This admin control ensures the system can be fine-tuned based on usage patterns and changing campus needs. The credit system effectively manages parking resources by preventing waste and ensuring fair access for all users while maintaining accountability through structured rules and penalties.

Objective 3: Enhance user experience through smart navigation and recommendations

The system implements comprehensive location-based features that significantly improve user convenience and navigation efficiency. All parking zones and individual slot locations are systematically collected and stored in the database with precise coordinates. This enables the system to guide users to their exact reserved slot locations. The integration of google maps on the home page allows users to visualize and check the location of each parking slot. Users can gain spatial awareness of the campus parking layout before making reservations. For users unfamiliar with the campus, the system offers intelligent recommendation functionality. Users can select their destination building and receive suggestions for optimal parking slots. The recommendation algorithm considers both driving time to reach the parking area and walking time from the slot to the selected destination building. This ensures the shortest total travel time and most convenient parking experience. This feature eliminates the guesswork involved in selecting appropriate parking locations and helps users make informed decisions based on their specific needs. After confirming a booking, users can utilize the navigation feature. The system provides turn-by-turn directions to lead the users directly to their reserved slot's exact location. This eliminates the common frustration of finding the general parking area but still struggling to locate the specific assigned space. Users enjoy a seamless end-to-end parking experience from reservation to arrival. This objective is successfully achieved through the combination of precise location data and intelligent recommendations and navigation that creates a user-friendly experience. The system minimizes confusion and reduces travel time while enhancing overall satisfaction with the parking system.

6.5 Concluding Remark

In this chapter, comprehensive testing conducted demonstrates that the parking reservation and management system has been successfully developed and validated against its core objectives. The system testing encompassed both performance evaluation and functional verification to ensure the application meets user requirements and operates reliably without error.

Performance testing revealed that the system delivers acceptable responsiveness with an average app start time of 1.11 seconds and stable database operations averaging 57.29ms response time. While the screen rendering performance concerns were identified with slow and frozen frames affecting approximately 20% of screens, the overall system performance remains within acceptable limits for practical deployment. The database performance testing confirmed that Firestore operations are efficient and stable to support the real-time interactions without noticeable delays to end users.

Furthermore, functional testing validated all core system components through systematic test cases covering user authentication, reservation management, smart recommendations, check-in processes, navigation features, administrative controls, and user profile management. All test cases passed successfully, confirming that the system handles both normal operations and exception scenarios appropriately. The testing demonstrated robust error handling, proper user feedback mechanisms, and seamless integration between different system components.

Based on the objective evaluation, the system has successfully achieved its three primary goals. The system effectively reduces traffic congestion through real-time slot monitoring and reservations, improves resource management through the comprehensive credit-based system with administrative oversight, and enhances user experience through intelligent recommendations and integrated navigation features.

Despite challenges related to hardware integration limitations and cost constraints that prevented full IoT implementation, the system successfully addresses the core parking management problems at UTAR Kampar campus through innovative workarounds. The QR code check-in system provides a practical solution that simulates real-time occupancy tracking by requiring users to physically scan codes at their reserved parking slots. This approach

maintains system integrity and ensures accurate slot status updates while remaining cost-effective and implementable without extensive hardware infrastructure.

The testing results validate that the developed solution provides a functional, reliable, and user-friendly parking management system that can significantly improve the current parking situation for campus users who drive. The QR code implementation creates a foundation that can be seamlessly integrated with IoT sensors in future enhancements, making the system scalable and adaptable to technological upgrades when resources permit. This design approach ensures immediate practical value while maintaining compatibility for future hardware-based real-time detection systems.

CHAPTER 7

Conclusion and Recommendations

7.1 Conclusion

The aim of this project was to improve parking management at UTAR Kampar campus by developing a comprehensive reservation and management system that addresses the critical challenges of traffic congestion and time wastage during peak hours. This objective has been successfully achieved through a multi-faceted technological solution that transforms how users interact with campus parking resources. The developed system effectively tackles the core problems by enabling users to check parking slot occupancy across campus zones in real-time and secure spaces before departure, eliminating the frustration of circling parking areas hoping to find available spots. The intelligent recommendation algorithm optimizes parking decisions by considering multiple factors including travel time and slot availability, while the integrated navigation system ensures users can efficiently locate their reserved spaces regardless of their familiarity with campus layout. The implementation of a credit-based management system ensures equitable resource distribution while preventing misuse through structured rules and penalties. Administrative oversight capabilities provide campus management with valuable insights and control mechanisms to optimize parking operations based on usage patterns and changing demands. While cost constraints necessitated simulation of real-time occupancy detection rather than full IoT integration, the QR code check-in system provides an effective intermediate solution that maintains system integrity and creates a scalable foundation for future hardware upgrades. This approach demonstrates that practical parking management improvements can be achieved without extensive infrastructure investment while remaining adaptable to technological enhancements. The comprehensive testing and evaluation confirm that all primary objectives were successfully met, validating the system's potential to significantly improve parking efficiency and user experience at UTAR Kampar campus. This project provides a valuable framework for institutions seeking cost-effective parking management solutions and demonstrates the feasibility of technology-driven approaches to campus infrastructure optimization.

7.2 Recommendation

To improve the system's functionality and make it user-friendly, there are some recommendations that could significantly improve the parking management solution. First, auto check-in functionality would significantly improve user convenience by automatically detecting when users arrive at their reserved slots through GPS proximity detection or plate recognition to eliminate the risk of users forgetting to check in and providing a seamless parking experience with automated push notifications.

Besides that, implementation of IoT hardware integration such as physical barrier systems with smart locks would ensure that only authorized users can access reserved spaces. Users can unlock barriers through the mobile app using Bluetooth or NFC communication to prevent unauthorized parking. Additionally, integrating ultrasonic or infrared occupancy sensors would provide real-time detection of actual slot usage to eliminate the reliance on user check-ins and provide accurate availability data. Smart lighting systems could guide users to their reserved slots through LED indicators.

The current system's major limitation lies in its static parking space configuration stored in the database, which lacks flexibility for campus layout changes. Another recommendation is to implement a dynamic parking space management interface for administrators that would allow real-time modification of parking layouts, adding or removing slots, changing zone configurations, and updating slot locations without requiring database restructuring through drag-and-drop map interfaces where administrators can visually modify parking layouts.

Additional enhancements could include integration with campus security systems for unauthorized parking detection, weather-based recommendations for covered or shaded parking, predictive analytics to forecast parking demand patterns, and integration with campus transportation systems to provide alternative suggestions such as shuttle services when parking demand exceeds availability. These improvements would transform the system from a basic reservation platform into a comprehensive smart parking ecosystem that adapts to changing campus needs while maximizing user convenience and resource efficiency.

REFERENCES

[1] B. Chougula, A. Tigadi, S. Jadhav, and G. Rudrappa, , "Automatic Smart Parking and Reservation System," *Bioscience Biotechnology Research Communications,* p. 107–113, 2020.

[2] H. Qin, N. Xu, Y. Zhang, Q. Pang, and Z. Lu, "Research on Parking Recommendation Methods Considering Travelers' Decision Behaviors and Psychological Characteristics," *Sustainability,* vol. 15, p. 6808, 2023.

[3] A. Aditya, S. Anwarul, R. Tanwar, and S. K. V. Koneru, "An IoT assisted Intelligent Parking System (IPS) for Smart Cities," *Procedia Computer Science,* vol. 218, pp. 1045-1054, 2023.

[4] H. Canli and S. Toklu, "AVL Based Settlement Algorithm and Reservation System for Smart Parking Systems in IoT-based Smart Cities," *The International Arab Journal of Information Technology,* p. 19, 2022.

[5] X. Zhang, K. Pitera, and Y. Wang, "Parking reservation techniques: A review of research topics, considerations, and optimization methods," *Journal of Traffic and Transportation Engineering (English Edition),* vol. 10, no. 2095-7564, pp. 1099-1117, 2023.

[6] A. Fahim, M. Hasan, and M. A. Chowdhury, "Smart parking systems: comprehensive review based on various aspects," *Heliyon,* vol. 7, no. 2405-8440, p. e07050, 2021.

[7] Ismail, Mohd and Jusoh, Muzammil and Sabapathy, Thennarasan and Osman, M.N. and A Rahim, Hasliza and Yasin, Najib and Mohd Fazilah, Ainur, "IoT Based Smart Parking System," *Journal of Physics: Conference Series,* vol. 1424, p. 012021, 2019.

[8] D. P. Carrasco, H. A. Rashwan, M. Á. García, and D. Puig, "T-YOLO: Tiny vehicle detection based on YOLO and multi-scale convolutional neural networks," *IEEE Access,* vol. PP, pp. 1-1, 2021.

[9] Parkalot, "Office Parking App, Sharing & Management Software System," Parkalot , [Online]. Available: https://parkalot.io/.

[10] FlexiParking, Flexi Parking™ - EASE YOUR PARKING EXPERIENCE, Flexi Parking.

[11] JomParking®, JomParking® - A quick and convenient way to pay for parking, JomParking®.

[12] Mouha, Radouan, "Internet of Things (IoT)," *Journal of Data Analysis and Information Processing,* vol. 09, pp. 77-101, 2021.

[13] T. C. Tsai and Y. Chen, "An IoT based parking recommendation system considering distance and parking lot flow," in *2021 International Conference on Information and Communication Technology Convergence (ICTC)*, 2021, pp. 978-983.

APPENDIX

Poster

UTAR
UNIVERSITI TUNKU ABDUL RAHMAN

Parking Reservation and Management System

for UTAR Kampar campus

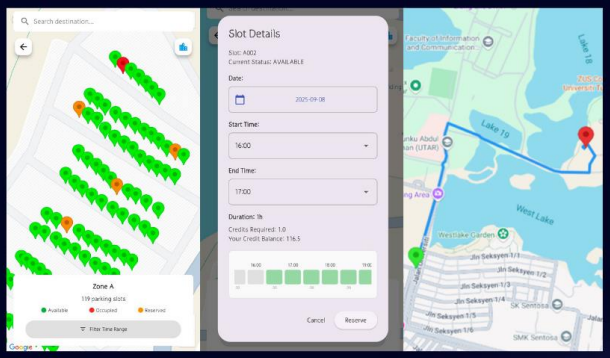
Introduction

Finding an available parking space during peak hours at UTAR Kampar campus is often time-consuming and frustrating. To address this issue, this project proposes the development of a mobile application that enables users to reserve parking slots in advance.

Methods

A mobile app was developed to enable advanced parking slot reservations using a credit-based system. Users receive monthly credits to book spaces in advance, reducing search time and uncertainty. The system provides real-time availability updates, intelligent parking recommendations, and navigation to reserved parking slots. Integration of the credit system ensure fair and efficient use of parking resources across the campus.

Result



- Advance Booking
- Real-Time Parking Availability
- Smart Recommendation
- Precise Slot Navigation
- Credit System

Discussion

The system's core features are designed to enhance parking efficiency on campus. By enabling advance reservations with a credit-based system, users spend less time searching for parking, easing traffic congestion and ensuring fair resource allocation. Smart recommendations and navigation further improve convenience, guiding drivers directly to their reserved slots for a smoother and more efficient parking experience.

Conclusion

The reservation system provides a smart parking solution for UTAR Kampar campus through advance booking, credit-based management, and integrated navigation. It enhances user convenience and optimizes campus parking resource utilization.

Developer: Kueh Wee Xin

Supervisor: Prof. Ts. Liew Soung Yue