

Parking Management with Mobile NFC Authentication

BY

NG SIN YEE

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION SYSTEMS (HONOURS)

INFORMATION SYSTEMS ENGINEERING

Faculty of Information and Communication Technology
(Kampar Campus)

FEBRUARY 2025

COPYRIGHT STATEMENT

© 2025 Ng Sin Yee. All rights reserved.

This Final Year Project report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Information Systems (Honours) Information Systems Engineering 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 sincere thanks and appreciation to my supervisors, Tan Lyk Yin who has given me this bright opportunity to engage in a Parking Management System with Mobile NFC Authentication. It is my first step to establish a career in information system field. A million thanks to you.

A special thanks to my sibling, Ng Kok Fung, for his advice, and for standing by my side during hard times. Finally, I must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course.

ABSTRACT

This project introduces a mobile application aimed at transforming residential parking management through the use of Near Field Communication (NFC) technology for secure and efficient vehicle authentication. Designed to meet the needs of residents, management, and visitors, the application offers a comprehensive solution to common parking challenges such as unauthorized access, poor space utilization, and lack of real-time monitoring. The key innovation of this system is its integration of NFC-based authentication. Vehicles are equipped with NFC tags containing specific data, which management can scan to verify parking authorization. If a tag fails to return valid data, it may indicate a counterfeit, allowing management to take immediate action and prevent unauthorized parking. This improves both security and transparency in the parking process. To support daily use, the application also enables residents to report illegal parking through a built-in reporting module and request temporary spots if their designated space is taken. For visitors, the system streamlines parking access: residents can reserve a spot and generate a unique reference code. Visitors enter this code into the application to receive their assigned parking lot, ensuring that only approved guests gain access. The management team benefits from a real-time overview of parking occupancy, allowing them to monitor space usage and respond promptly to reported issues. A visual floor plan aids visitors in navigating to their assigned spots, further enhancing usability. Unlike existing solutions, which often address only a portion of the parking management process, this application provides a fully integrated system that combines secure vehicle verification, resident engagement, visitor access control, and real-time administrative tools. The objective is to develop a reliable, intuitive mobile application that leverages NFC for secure sticker validation, facilitates efficient space allocation, and improves communication among users. By enhancing both the security and convenience of parking management, this solution aims to reduce unauthorized parking incidents, optimize space usage, and provide a better experience for all stakeholders.

Area of Study (Minimum 1 and Maximum 2): Mobile Applications, Wireless Communication Technology

Keywords (Minimum 5 and Maximum 10): NFC Authentication, Parking Management, Mobile Application, Visitor Access, Real-Time Monitoring, Illegal Parking Reporting, Resident Management, Parking Allocation, Secure Vehicle Verification

TABLE OF CONTENTS

| | |
|---|------------|
| COPYRIGHT STATEMENT | ii |
| ABSTRACT | iv |
| TABLE OF CONTENTS | v |
| List of Figures..... | iv |
| List of Table | vii |
| List of Abbreviations..... | ix |
| | |
| CHAPTER 1 | 1 |
| 1.1 Project Background | 1 |
| 1.2 Problem Statement and Motivation | 2 |
| 1.3 Project Objectives..... | 5 |
| 1.4 Project Contribution | 6 |
| 1.5 Project Scope | 7 |
| 1.6 Report Organization..... | 9 |
| | |
| CHAPTER 2 | 11 |
| 2.1 Authentication in Parking Systems..... | 11 |
| 2.1.1 Comparative Field Study on Residential Parking Practices | 11 |
| 2.1.2 Current Trends in Parking System Technologies | 16 |
| 2.2 NFC-Based Authentication..... | 19 |
| 2.2.1 Case Studies of NFC-Based Authentication..... | 19 |
| 2.2.2 Comparison of NFC With Other Authentication Methods | 21 |
| 2.3 Mobile Application Development | 23 |
| 2.3.1 Case studies on Parking Systems using Mobile Application..... | 23 |
| 2.3.2 Importance of user-friendly in Mobile Application Development | 29 |
| 2.4 Proposed Residential Parking Management System with Mobile NFC Authentication..... | 30 |
| | |
| CHAPTER 3 | 33 |
| 3.1 System Development Models..... | 33 |
| 3.1.1 System Development Model 1: Waterfall | 33 |
| 3.1.2 System Development Model 2: Agile | 34 |
| 3.1.3 System Development Model 3: Prototype | 35 |
| 3.1.4 Selected Model: Prototype Model..... | 37 |

| | | |
|----------------------|--|-----------|
| 3.2 | System Requirements | 38 |
| 3.2.1 | Hardware Components | 38 |
| 3.2.2 | Software Components | 40 |
| 3.3 | Functional Requirements | 41 |
| 3.4 | Project Milestones | 45 |
| 3.4.1 | Project 1 Timeline | 45 |
| 3.4.2 | Project 2 Timeline | 47 |
| 3.5 | Estimated Cost | 49 |
| 3.6 | Concluding Remark | 50 |
| CHAPTER 4 | | 52 |
| System Design | | 52 |
| 4.1 | System Architecture | 52 |
| 4.2 | Use Case Diagram | 54 |
| 4.3 | Activity Diagram..... | 57 |
| 4.3.1 | Login Activity Diagram | 57 |
| 4.3.2 | Sign Up Activity Diagram..... | 58 |
| 4.3.3 | Forgot Password Activity Diagram..... | 60 |
| 4.3.4 | Visitor Main Page..... | 62 |
| 4.3.5 | Auto Assignment Parking Activity Diagram..... | 64 |
| 4.3.6 | Report Illegal Parking | 66 |
| 4.3.7 | View My Reports | 68 |
| 4.3.8 | Register Visitor Activity Diagram..... | 70 |
| 4.3.9 | Manage Illegal Parking | 73 |
| 4.3.10 | NFC Tag Writer..... | 74 |
| 4.3.10 | NFC Tag Reader..... | 77 |
| 4.4 | Database Design..... | 79 |
| 4.5 | Concluding Remark | 84 |
| CHAPTER 5 | | 85 |
| 5.1 | Software Setup and Configuration | 85 |
| 5.2 | Implementation Details..... | 87 |
| 5.2.1 | Implementation Details for Firebase Integration..... | 87 |
| 5.3 | System Operation | 98 |
| 5.3.1 | Login Module..... | 98 |
| 5.3.2 | Sign up module | 99 |

| | |
|---|------------|
| 5.3.3 Forgot Password..... | 100 |
| 5.3.4 Visitor Module | 101 |
| 5.3.5 Carpark Spot Auto Assignment..... | 102 |
| 5.3.6 View Map Module | 103 |
| 5.3.7 Resident Main Module..... | 104 |
| 5.3.8 Report Illegal Parking Module..... | 105 |
| 5.3.9 Register Visitor Module | 106 |
| 5.3.10 Visitor Reference Code Module..... | 107 |
| 5.3.11 View My Reports Module | 108 |
| 5.3.12 Management Main Page..... | 109 |
| 5.3.13 Manage Visitor Parking Module | 110 |
| 5.3.14 View Illegal Parking Module | 111 |
| 5.3.15 NFC Car Details Module | 112 |
| 5.3.16 NFC Tag Writer Module | 113 |
| 5.3.17 NFC Tag Reader..... | 114 |
| 5.4 Concluding Remark | 116 |
| CHAPTER 6..... | 117 |
| System Evaluation and Discussion | 117 |
| 6.1 System Performance Evaluation | 117 |
| 6.1.1 Loading Test..... | 117 |
| 6.1.2 Response Time Analysis | 118 |
| 6.1.3 Network Performance | 119 |
| 6.1.4 Database Performance | 120 |
| 6.2 System Setting, Setup and Result | 123 |
| 6.3 Project Challenge..... | 132 |
| 6.4 Objective Evaluation..... | 133 |
| 6.5 Concluding Remark | 135 |
| CHAPTER 7..... | 136 |
| 7.1 Conclusion | 136 |
| 7.2 Recommendation/ Future Work | 137 |
| REFERENCES..... | 138 |
| POSTER | 141 |

List of Figures

| Figure Number | Title | Page |
|----------------------|---|-------------|
| Figure 2.1.1.1 | Unisuite visitor logbook | 12 |
| Figure 2.1.1.2 | Unisuite Carpark Sticker | 12 |
| Figure 2.1.1.3 | Unisuite Resident reserve the parking with a paper. | 12 |
| Figure 2.1.1.4 | Champs Élysées Resident Carpark Sticker | 13 |
| Figure 2.1.1.5 | The Meadow Park Visitor Verification Paper | 14 |
| Figure 2.1.1.6 | The Meadow Park Resident Carpark Sticker | 14 |
| Figure 2.1.1.7 | The Meadow Park Visitor Logbook | 15 |
| Figure2.1.2.1 | LRP technology from ZKTeco Malaysia | 17 |
| Figure2.1.2.2 | LRP technology access methods from ZKTeco Malaysia | 18 |
| Figure2.1.2.3 | Parking Equipment from timeTec | 18 |
| Figure2.1.2.4 | QR code technology from Veemios | 19 |
| Figure 2.3.1.1 | Parking Management System for Veemios | 25 |
| Figure 2.3.1.2 | Parking Management System for i-Neighbour | 27 |
| Figure 2.3.1.3 | Sunway Smart Parking | 28 |
| Figure 3.1.1 | Waterfall Model | 34 |
| Figure 3.1.2 | Agile Model | 35 |
| Figure 3.1.3 | Prototype Model | 36 |
| Figure 3.2.2.1 | Android Studio | 40 |
| Figure 3.2.2.2 | Firebase | 41 |
| Figure 3.4.1.1 | Gantt Chart of Project 1 Timeline | 44 |
| Figure 3.4.2.1 | Gantt Chart of Project 2 Timeline | 46 |
| Figure 4.1.1 | Block Diagram | 51 |

| | | |
|----------------|---|----|
| Figure 4.2.1 | Use Case Diagram | 53 |
| Figure 4.3.1 | Login Activity Diagram | 57 |
| Figure 4.3.2 | Sign Up Activity Diagram | 59 |
| Figure 4.3.3 | Forgot Password Activity Diagram | 61 |
| Figure 4.3.4 | Visitor Main Activity Diagram | 63 |
| Figure 4.3.5 | Auto Assignment Parking Activity Diagram | 65 |
| Figure 4.3.6 | Report Illegal Parking Activity Diagram | 67 |
| Figure 4.3.7 | View My Reports Activity Diagram | 69 |
| Figure 4.3.8 | Register Visitor Activity Diagram | 71 |
| Figure 4.3.9 | Manage Illegal Parking Activity Diagram | 73 |
| Figure 4.3.10 | NFC Tag Writer Activity Diagram | 75 |
| Figure 4.3.11 | NFC Tag Reader Activity Diagram | 77 |
| Figure 4.4.1 | Database Design | 78 |
| Figure 5.1.1 | Android Studio Official Website | 85 |
| Figure 5.1.2 | Firebase Website | 86 |
| Figure 5.2.1.1 | Add the Firebase Authentication SDK Dependencies to build.gradle | 87 |
| Figure 5.2.1.2 | Sync Project with Gradle files. | 87 |
| Figure 5.2.1.3 | Enable Authentication Methods in Firebase Console | 88 |
| Figure 5.2.1.4 | Add Firestore dependencies in build.gradle | 91 |
| Figure 5.2.1.5 | Sync the project with Gradle Files | 91 |
| Figure 5.2.1.6 | Set Up Firestore Database | 92 |
| Figure 5.2.1.7 | Add Firebase Storage SDK dependencies in build.gradle | 93 |
| Figure 5.2.1.8 | Sync project with Gradle Files | 94 |
| Figure 5.2.1.9 | Set up Firebase Storage | 94 |

| | | |
|-----------------|--|-----|
| Figure 5.3.1 | Login Module | 97 |
| Figure 5.3.2 | Sign Up Module | 98 |
| Figure 5.3.3 | Forgot Password Module | 99 |
| Figure 5.3.4 | Visitor Module | 100 |
| Figure 5.3.5 | Carpark Spot Auto Assignment Module | 101 |
| Figure 5.3.6 | View Map Module | 102 |
| Figure 5.3.7 | Resident Main Module | 103 |
| Figure 5.3.8 | Report Illegal Parking Module | 104 |
| Figure 5.3.9 | Register Visitor Module | 105 |
| Figure 5.3.10 | Visitor Reference Code Module | 106 |
| Figure 5.3.11 | View My Reports Module | 107 |
| Figure 5.3.12 | Management Main Module | 108 |
| Figure 5.3.13 | Manage Visitor Parking Module | 109 |
| Figure 5.3.14.1 | Illegal Parking Report | 110 |
| Figure 5.3.14.2 | Illegal Parking Notification | 110 |
| Figure 5.3.15 | NFC Car Details Module | 111 |
| Figure 5.3.16.1 | NFC Tag Writer Module | 112 |
| Figure 5.3.16.2 | NFC Tag Write Successfully | 113 |
| Figure 5.3.17.1 | NFC Tag Reader Module | 114 |
| Figure 5.3.17.2 | NFC Tag Read Successfully | 114 |
| Figure 6.1.1 | Loading Test for report image | 116 |
| Figure 6.1.2 | Response Time Analysis for Auto Assignment Parking | 118 |
| Figure 6.1.3 | Network Performance for Image Upload to Database | 119 |
| Figure 6.1.4 | Database Performance for CSV file updated the Database | 120 |

List of Table

| Table Number | Title | Page |
|---------------------|--|-------------|
| Table 2.1.1.1 | Summary of Three Studied Residential Areas | 15 |
| Table 2.2.1 | Summary of Comparison of NFC With Other Authentication Methods | 23 |
| Table 2.4.1 | Comparison Table of the Proposed Residential Parking Management System and Reviewed Parking Management Systems | 31 |
| Table 3.2.1.1 | Smartphone, NFC Tags and Laptop Specification | 39 |
| Table 3.2.2.1 | Minimum Requirements to Install Android Studio | 40 |
| Table 3.2.2.2 | Minimum Requirements to Use Firebase | 41 |
| Table 3.5.1 | Estimated Cost Calculation | 48 |
| Table 4.4.1 | Resident Table | 79 |
| Table 4.4.2 | Management Table | 79 |
| Table 4.4.3 | Visitor Table | 80 |
| Table 4.4.4 | Parking Table | 81 |
| Table 4.4.5 | Illegal Report Table | 81 |
| Table 4.4.6 | Nfc_Carinfo Table | 82 |
| Table 5.2.1.1 | Example to use Firebase Authentication to handle user register | 89 |
| Table 5.2.1.2 | Example to use Firestore to Store Visitor Details | 92 |
| Table 5.2.1.3 | | 95 |

| | | |
|--------------|--|-----|
| | Example to use Firebase Storage to Store Illegal Parking | |
| Table 6.2.1 | Photo | 122 |
| Table 6.2.2 | Sign Up Test Case | 123 |
| Table 6.2.3 | Login Test Case | 124 |
| Table 6.2.4 | Forgot Password Test Case | 124 |
| Table 6.2.5 | Visitor Module Test Case | 125 |
| Table 6.2.6 | Parking Assignment Test Case | 126 |
| Table 6.2.7 | Report Illegal Parking Test Case | 126 |
| Table 6.2.8 | Register Visitor Test Case | 127 |
| Table 6.2.9 | View My Reports Test Case | 128 |
| Table 6.2.10 | Manage Visitor Parking Test Case | 128 |
| Table 6.2.11 | Illegal Parking Report Test Case | 129 |
| Table 6.2.12 | NFC Car Details Test Case | 129 |
| Table 6.2.13 | NFC Tag Writer Test Case | 130 |
| | NFC Tag Reader Test Case | |

List of Abbreviations

| | |
|------|-------------------------------------|
| NFC | Near Field Communication |
| PPR | People's Housing Project |
| LPR | License Plate Recognition |
| UHF | Ultra-High Frequency |
| RFID | Radio Frequency Identification |
| ALRP | Automated License Plate Recognition |
| RF | Radio Frequency |
| URL | Uniform Resource Locator |
| QR | Quick Response |
| URL | Uniform Resource Identifier |

CHAPTER 1

Introduction

1.1 Project Background

The term "Near Field Communication" (NFC) refers to a group of short-range wireless communication technologies that allow data transfer between devices at a just few inches apart. NFC is a popular option for applications for rapid and secure interactions because it enables safe, contactless transactions and data exchanges.[1] NFC is derived from radio frequency identification (RFID) technology. When NFC was first introduced in the early 2000s, it was widely adopted in a number of industries, such as access control systems, public transportation ticketing, and mobile payments. This technology is perfect for applications that require simplicity and security because of its ability to create a smooth, user-friendly experience through straightforward, touch-based interactions. It has completely changed the way people handle and validate information.

Mobile phones were originally designed for voice communication, but now mobile phones have become an indispensable device in life. Today's smartphones are equipped with many tools such as advanced processors, high-resolution cameras and powerful connectivity options including NFC. The popularity of smartphones has diversified the use of technology, such as navigation, online shopping, and many applications with different functions. This brings many conveniences as well as solutions to various challenges including to parking management.

The shortage of parking spaces for private cars in the People's Housing Project (PPR) areas of Kuala Lumpur has become a pressing issue, leading to dissatisfaction among residents and the public.[2] According to a recent report, the limited availability of parking spaces, coupled with inadequate enforcement in free parking zones, exacerbates the problem, creating significant frustration. Property owners often take advantage of the lack of regulation, further intensifying the parking crisis. In this urban setting, where space is at a premium, inefficient parking management has emerged as a

critical concern, highlighting the need for a more organized and effective solution to manage parking resources and enforce regulations.

This context underscores the importance of developing innovative solutions to tackle parking issues effectively. Our project aims to address these challenges by leveraging NFC technology to create a mobile application that enhances parking control and safety. By integrating cutting-edge technology with practical management tools, the project aim to provide a comprehensive solution that addresses the needs of residents, management departments, and visitors, ultimately contributing to a more organized and efficient parking system.

1.2 Problem Statement and Motivation

The persistent parking problem in Kuala Lumpur's residential areas, especially in apartment buildings poses a huge challenge to residents and local authorities. According to The Sun, there is a severe shortage of parking spaces due to insufficient enforcement. For example, abandoned vehicles take up valuable parking spaces, causing increasing frustration among residents who must search for available parking spaces in congested lots [3]. Additionally, according to The Star, disputes between neighbours over parking spaces have become commonplace, highlighting the impact poor parking management has on community relations [4]. These problems highlight the urgent need for a comprehensive parking management system to alleviate the shortage of parking spaces, reduce disputes, and improve the overall efficiency of community parking spaces through effective management.

1. Inadequate Parking Security Measures

Current community parking management systems have serious security flaws due to their reliance on outdated verification methods. The main problem is the use of handwritten vehicle stickers and serial numbers, which can easily be counterfeited and tampered with. This manual way of recording vehicle information creates opportunities

for fraudulent activity, such as creating fake parking stickers, which can result in unauthorized vehicles occupying designated parking spaces.

In addition, security is tasked with verifying these handwritten records, a process that is not only labour-intensive, but also error-prone and inefficient. Relying on paper records for identity verification means there is no effective mechanism to ensure the purpose of the visitor's visit. Thus, making it easy for unauthorized individuals to bypass the system and gain access to restricted parking areas. The lack of strong security measures ultimately compromises the effectiveness of parking management and results in unauthorized use of parking spaces.

This is the motivation to develop a parking management system with NFC authentication. By implementing an NFC-based authenticated parking management system, the application can significantly enhance security, reduce the risk of fraud and streamline the verification process. This transformation will not only ensure the rational use of parking spaces, but also improve the overall efficiency and effectiveness of parking management.

2. Limited Resident and Visitor Parking Visibility

Existing residential parking management systems often provide insufficient information to residents and visitors regarding parking space availability and location. As a result, residents or visitor may misallocation of parking space and park in the wrong parking space leading to conflicts and unauthorized use of designated spots. Also, visitor may be unable to find the corresponding parking space and this will increase frustration and time wastage. This caused a lot of trouble and reduced efficiency.

This is the motivation to develop a mobile application that offers comprehensive visibility into parking layouts for residents and visitors is crucial for optimizing parking utilization. By offering a user-friendly interface, parking location displays will assist residents and visitor in finding their assigned or available parking spaces promptly. This

enhanced visibility reduces the frustration and wastes time of finding parking spaces, increases overall parking efficiency, and minimizes the potential for unauthorized parking and conflicts.

3. Inefficient Reporting and Enforcement Mechanisms

Illegal parking problems in residential areas often lead to serious problems, such as blocked access, reduced parking spaces, occupied parking spaces, and increased resident dissatisfaction. When unauthorized vehicles occupy designated parking spaces or park in prohibited areas, it disrupts parking and disrupts parking.

Currently, reporting illegal parking usually requires residents to contact management through cumbersome methods such as phone calls or in-person visits. These traditional methods are inefficient and can cause delays in resolving breaches. Additionally, the lack of a centralized reporting system means reports are often scattered across different communication channels, making it difficult for management to track and address incidents in a timely manner. This fragmented process hinders effective enforcement and resolution of parking issues.

This motivated to develop parking management mobile application with report illegal parking function. The application will allow residents to easily report illegal parking directly from their smartphones, streamlining the process and ensuring management receives real-time notifications. At the same time, residents whose parking spaces are occupied can park their cars in the temporary parking lot to avoid traffic obstruction. This will make enforcement operations more effective and improve overall control of parking violations.

1.3 Project Objectives

1. To implement Secure NFC Authentication in a parking management system mobile application using Android Studio.

To enhance security and efficiency in parking management, this project will be implementing NFC technology into our mobile application using Android Studio. The NFC technology will be integrated by attaching NFC chips to parking stickers, which will be issued to authorized vehicles. The mobile application will be equipped with an NFC reader API, allowing it to securely scan and retrieve data from these chips. This process involves developing the NFC reading functionality within the Android Studio. This is ensuring that it can accurately authenticate parking stickers and verify their legitimacy. By integrating NFC technology, the project aim to prevent unauthorized access and illegal duplication of parking stickers, significantly improving security within parking facilities. This goal is to prevent unauthorized access, or illegal copying of parking stickers and ensure higher levels of security within parking facilities.

2. To Develop an interface that simplifies information entry and access to dedicated floor plans.

To create an interface in the mobile application that simplifies information entry and easy access to floor plans, this project will transition from traditional manual methods to a more efficient digital process. Traditionally, residents and visitors may have relied on physical or verbal communication to learn parking details and find parking spaces. This approach can be cumbersome and error prone.

The proposed method is developing a parking management mobile application that automatically assigns available parking space to the user when the visitor or resident needs parking. Also, a floor plan is provided for navigation. This eliminates manual paperwork, reduces errors, and makes navigation more efficient.

3. To create a dashboard in the mobile application for monitoring occupancy and verifying parking stickers via NFC

To develop a comprehensive management dashboard in the mobile application, this project will integrate features for enhanced parking oversight and control. The dashboard will provide parking space occupancy, a feature that allows managers to see which parking spaces are available or occupied. It will also include NFC verification capabilities, allowing managers to scan and confirm the authenticity of parking stickers using their mobile devices, preventing unauthorized access as well as counterfeit parking stickers. Additionally, the dashboard will feature a system for managing illegal parking reports from residents. Aggregating these reports allows for effective tracking and timely response. This goal focuses on providing management with effective tools to enforce parking regulations, optimize parking space utilization, and resolve issues quickly to ensure a more organized and safe parking environment.

1.4 Project Contribution

The contribution of this project lies in its innovative approach to parking management, integrating NFC technology to address key challenges and enhance various aspects of the parking experience.

1. Enhanced Security Measures

The NFC authentication system enhance parking security by assisting in verifying authorized vehicles. While it does not directly ensure that only authorized vehicles occupy designated parking spaces, it simplifies the verification process by allowing management to confirm the identity and information of vehicles and their owners through NFC-enabled parking stickers. The system makes it easier to identify unauthorized vehicles and monitor parking space occupancy. By improving the accuracy and efficiency of vehicle verification, the system contributes to a more secure and well-regulated parking environment.

2. Community Engagement and Collaboration

The project fosters community engagement by involving residents in the parking management process. The ability for residents to report instances of illegal parking promotes a collaborative approach to maintaining order within the community. By creating a platform that encourages residents to actively participate in parking regulation enforcement, the project establishes a sense of shared responsibility and community well-being.

3. Visitor Convenience

The project's contribution extends to visitors by providing them with a simplified and convenient parking experience. Through the application, visitors can input their information for parking authorization and easily navigate designated parking areas. This targeted approach ensures that visitors can swiftly locate their parking spaces, contributing to a positive overall user experience for both residents and visitors.

In summary, the project's contribution lies in its holistic approach to parking management, leveraging NFC technology to improve efficiency, enhance security measures, promote community engagement, and provide a seamless experience for both residents and visitors.

1.5 Project Scope

The project involves developing a mobile application to enhance the security and efficiency of parking management within residential communities. The core focus is the integration of NFC technology for secure authentication, enabling management to verify parking stickers, confirm vehicle identity, and ensure that only authorized vehicles access designated parking spaces. This NFC-based verification system will allow management to swiftly address resident reports of illegal parking, promoting a secure and well-regulated parking environment.

Residents will benefit from several features designed to streamline their parking experience. Through the application, residents can reserve visitor parking spaces, which generates a unique reference code for their guests. This code ensures that visitors are pre-authorized and that they are parking for legitimate purposes. If a resident's designated parking spot is occupied, the system will automatically assign a temporary parking space, reducing inconvenience and preventing road blockages. Additionally, the application provides a robust reporting mechanism, allowing residents to report instances of illegal parking directly through the application. These reports are sent to management in real-time, ensuring timely review and response.

For visitors, the application simplifies the parking process by offering an intuitive interface for entering details, including the reference code provided by the resident. The system will automatically assign a parking spot to the visitor and provide access to a simplified floor plan that highlights visitor parking areas, making navigation easy and efficient.

1.6 Report Organization

This report is organized into seven comprehensive chapters that document the development lifecycle of the Parking Management System with NFC Authentication, from the conceptualization of the idea to system evaluation and conclusion. Each chapter plays a critical role in portraying the technical, functional, and managerial aspects of the system.

Chapter 1: Introduction

This chapter provides an overview of the project, including the background and the need for a smarter parking management system, especially in residential and institutional settings. It highlights the problems faced with traditional parking systems and explains how integrating NFC authentication can enhance security and user convenience. The chapter also outlines the objectives, contributions, and scope of the system, followed by this section that details how the rest of the report is organized.

Chapter 2: Literature Review

This chapter explores existing parking management systems, particularly those incorporating technologies such as RFID, QR code, or NFC. It analyses the strengths and weaknesses of these systems, laying the foundation for why an NFC-based solution is both innovative and practical for modern environments.

Chapter 3: System Methodology

This chapter describes the development methodology adopted for building the system. It explains why the Prototype Model was chosen, considering the need for iterative testing and user feedback. It also presents the system and functional requirements, the project timeline for both Project I and Project II, and an estimated cost section. The tools and software used such as Android Studio, Firebase, and NFC-compatible devices are also discussed here.

Chapter 4: System Design

This chapter focuses on the technical architecture and visual representation of the system. It includes the overall system architecture, use case diagrams to define user roles (Resident, Management, Visitor), activity diagrams, and the structure of the Firestore database used to store user and visitor data. This chapter helps readers understand how the system components interact and how data flows within the application.

Chapter 5: System Implementation

This chapter covers the setup and configuration of the software environment, such as setting up Firebase and implementing NFC authentication in Android. It also includes details on how the application was implemented for different roles (Resident, Management, Visitor), and highlights the core modules such as visitor registration, resident authentication via NFC, and real-time data access for management. Any issues encountered during implementation and how they were resolved are also discussed.

Chapter 6: System Evaluation and Discussion

This chapter evaluates the system's performance through testing scenarios that simulate real-world usage. It includes testing results of NFC functionality, system responsiveness, and user experience for each user role (Resident, Management, Visitor). The chapter also discusses whether the project has successfully achieved its original objectives, with evidence based on the performance and functionality of the system.

Chapter 7: Conclusion and Recommendation

The final chapter summarizes the accomplishments of the project and reflects on its impact. It offers suggestions for future work, such as integrating license plate recognition or expanding the system for commercial use. The conclusion reinforces the value of NFC technology in secure, efficient parking management.

CHAPTER 2

LITERATURE REVIEW

2.1 Authentication in Parking Systems

2.1.1 Comparative Field Study on Residential Parking Practices

This project conducted on-site study of three residential areas located in Kampar which are Unisuite, Champs Élysées, and The Meadow Park. This on-site review conducted to understand the traditional parking system methods and their limitations.

At Unisuite, visitors are allowed entry by simply filling out their details in a provided logbook without any verification of their identity. The security guards do not ask for any unit numbers or contact the residents to confirm the visitor's legitimacy. For resident, they can use the access card to enter. Unisuite issues traditional car park stickers for resident vehicles, where the series number is printed, but the vehicle carplate number and parking lot number are handwritten. This practice raises concerns about potential tampering or the creation of counterfeit stickers. Additionally, residents sometimes hang papers on their parking spots indicating which vehicle should occupy that space. However, since no one verifies the authenticity of these papers, it can lead to parking space misuse.

In cases where a resident's parking slot is occupied by another vehicle, the resident must go to the management office to report the illegal parking. The management then has to notify the security guard to take actions like clamping the car and issuing a fine notice. After this series of actions, the resident is assigned a temporary parking space. This process is time-consuming and inconvenient. Furthermore, verifying details becomes difficult for management as the handwritten information on the stickers fades over time. Management's reliance on paper documentation for recording vehicle details and corresponding series numbers adds to their workload, as they need to search through physical records to verify the information.

CHAPTER 2

[illegible]

Figure 2.1.1.1 Unisuite visitor logbook

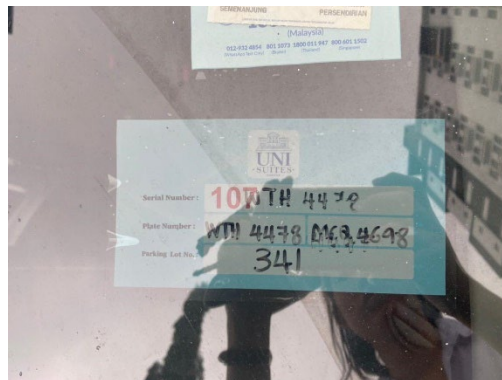


Figure 2.1.1.2 Unisuite Carpark Sticker



Figure 2.1.1.3 Unisuite Resident reserve the parking with a paper.

Champs Élysées does not provide any visitor parking. Visitors must park in external public parking areas and then walk to the lobby. Before entering the lobby, visitors must fill out their personal details, then security guard will use their access card to open the lobby door. For resident parking, resident can access the carpark by the access card. Each vehicle is issued a Champs Élysées management car park sticker, with a printed series number but a handwritten vehicle number and parking lot number which similar to Unisuite. This method also carries the risk of sticker tampering or counterfeiting. The challenges in verifying vehicle details are the same, as the handwritten numbers fade over time, making it difficult for management to check the authenticity of the stickers. The process for dealing with illegal parking is equally cumbersome, residents must report the issue to management, who will then instruct security to take action like clamping the car and issuing a fine notice, and finally, assign a temporary parking space to the affected resident. This process is time-consuming and inconvenient.



Figure 2.1.1.4 Champs Élysées Resident Carpark Sticker

The Meadow Park allows visitor parking after verifying the visitor's identity against the information the visitor fill in the logbook. Once verified, the visitor is assigned a parking space, and a paper indicating they are a visitor is placed on the vehicle dashboard for security checks during patrols. Visitors must return this paper upon exit. For residents, residents can use the access card to enter The Meadow Park. The Meadow Park management issues a car park sticker with the parking lot number and vehicle license plate handwritten, and printed series number just like the other two residential

areas. Over time, the information on these stickers' fades, especially for outdoor parking, it is complicating for verification. Management uses paper documentation to record resident details, vehicle numbers, and parking series numbers, and when checks are needed, they have to search through these documents. This may lead a time-consuming and inefficient process. If a resident's parking space is occupied, they must report the issue at the guard house, after which the security will clamp the offending car, issue a fine notice, and then assign a temporary parking space to the resident.

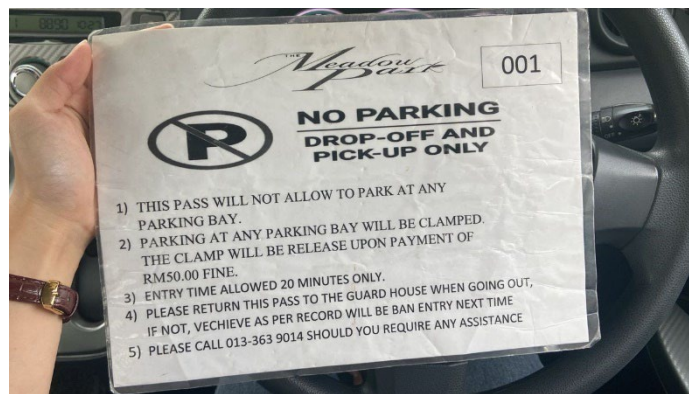


Figure 2.1.1.5 The Meadow Park Visitor Verification Paper



Figure 2.1.1.6 The Meadow Park Resident Carpark Sticker

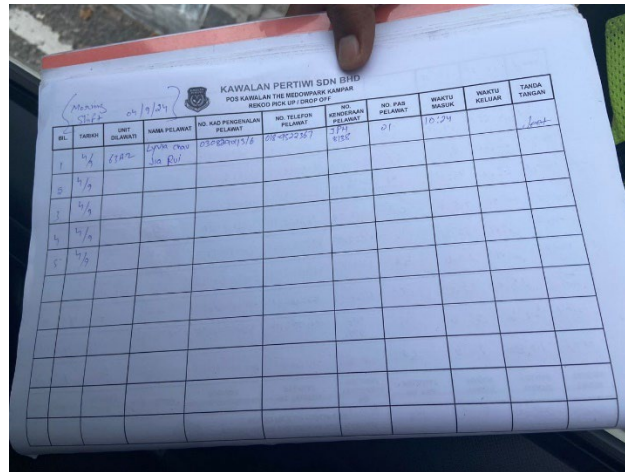


Figure 2.1.1.6 The Meadow Park Visitor Logbook

These observations highlight the inefficiencies and potential security risks associated with traditional parking management methods, such as manual logbooks, handwritten parking stickers, and the reliance on paper documentation.

| Aspect | Unisuite | Champs Élysées | The Meadow Park |
|------------------|--|--|--|
| Visitor Entry | Fill in a logbook; no identity verification; no verify with residents | Fill in a logbook; no identity verification; no verify with residents | Fill in a logbook; identity verified; no verify with residents |
| Visitor Parking | Allowed; a visitor parking space is assigned | Not allowed; visitors park in public spaces outside | Allowed; a visitor parking space is assigned |
| Resident Parking | Car park sticker with printed series number; handwritten vehicle and parking lot numbers | Car park sticker with printed series number; handwritten vehicle and parking lot numbers | Car park sticker with printed series number; handwritten vehicle and parking lot numbers |

| | | | |
|---------------------------------------|--|--|--|
| Potential for Tampering | High - due to handwritten details on stickers | High - due to handwritten details on stickers | High - due to handwritten details on stickers |
| Verification Car Park Sticker Process | Time-consuming; paper documentation; handwritten details fade over time | Time-consuming; paper documentation; handwritten details fade over time | Time-consuming; paper documentation; handwritten details fade over time |
| Handling Illegal Parking | visit to management office to make report; security guard takes action; temporary parking assigned | visit to management office to make report; security guard takes action; temporary parking assigned | visit to management office to make report; security guard takes action; temporary parking assigned |

Table 2.1.1.1 Summary of Three Studied Residential Areas

2.1.2 Current Trends in Parking System Technologies

The rapid evolution of parking system technologies has significantly enhanced the efficiency, security, and convenience of parking management across various residential and commercial settings. This literature review explores the current trends in parking system technology, focusing on License Plate Recognition (LPR), Ultra High Frequency (UHF) and Radio-Frequency Identification (RFID) systems, and QR code.

LPR technology uses cameras to capture and recognize vehicle license plates. LPR technology has rapidly become widespread in various commercial settings due to its ability to automate vehicle identification and access control. The LPR system streamlines the entry and exit process by automatically capturing and identifying vehicle license plates, eliminating the need for manual inspections or traditional parking tickets. In locations where parking fees are required, users can pay online via an

application and exit the parking lot without needing to locate a kiosk or use cash to pay for a ticket. LPR systems are typically integrated with barrier gates to enhance security and reduce the risk of unauthorized access. According to ZKTeco Malaysia, their LPR system achieves an accuracy of 99.9% during the day and 99.7% at night, with an adaptable speed of 15 km/h and a reading distance of 10 m. These systems not only improve the accuracy of vehicle tracking but also contribute to a more seamless parking experience for users [5].

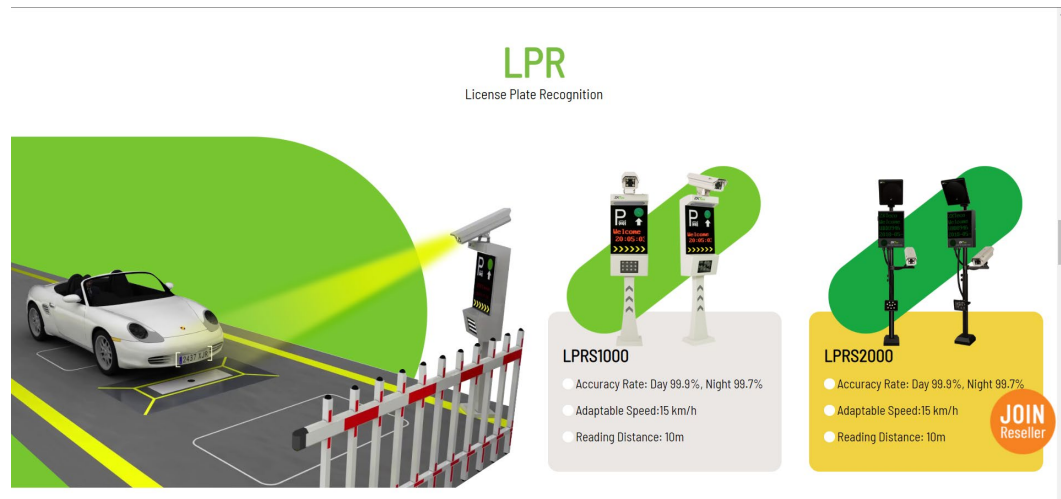


Figure2.1.2.1 LRP technology from ZKTeco Malaysia

Ultra-High Frequency (UHF) and Radio Frequency Identification (RFID) **systems** are the primary systems for parking management, especially in residential areas where access control is a priority. According to ZKTeco Malaysia, these systems allow residents to enter parking facilities using UHF cards, Anti-Torn Stickers, UHF Windshield Tags, and Heavy-Duty UHF Tags. [5] These access methods are pre-programmed with the resident's vehicle and parking details, enabling quick and efficient access. Different access methods can be used in various scenarios; for instance, residents can place an Anti-Torn Sticker or UHF Windshield Tag on their vehicle's windshield, allowing for easy access without needing to find and scan a UHF card while driving. UHF or RFID readers can directly read the Anti-Torn Sticker and UHF Windshield Tag attached to the windshield. The Heavy-Duty UHF Tag is implemented on the car plate and used similarly to the Anti-Torn Sticker and UHF Windshield Tag. By integrating UHF/RFID technology into barrier gate systems, residents can easily

access parking areas without physical interaction with security personnel, thereby enhancing security and preventing unauthorized vehicles from entering. According to timeTec, their UHF or RFID Reader has a reading distance of 20 m [6].

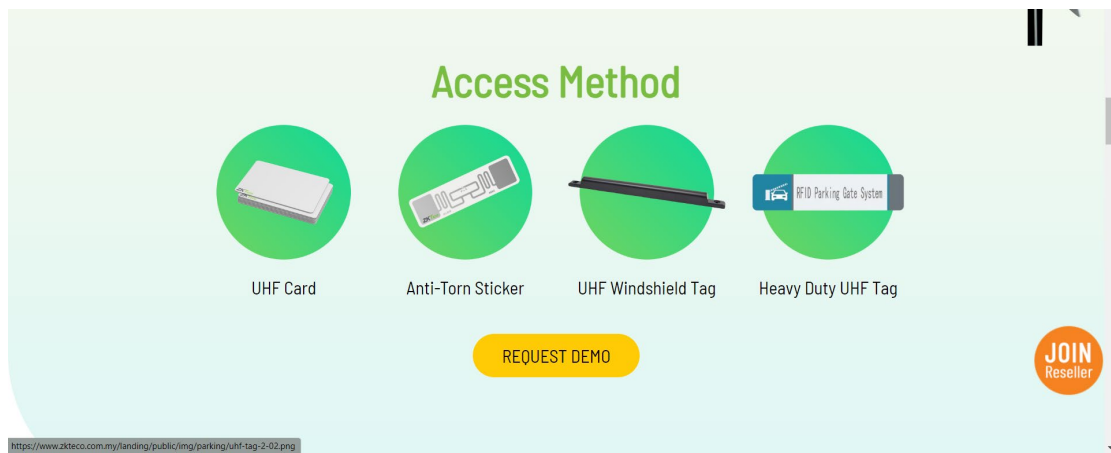


Figure2.1.2.2 LRP technology access methods from ZKTeco Malaysia

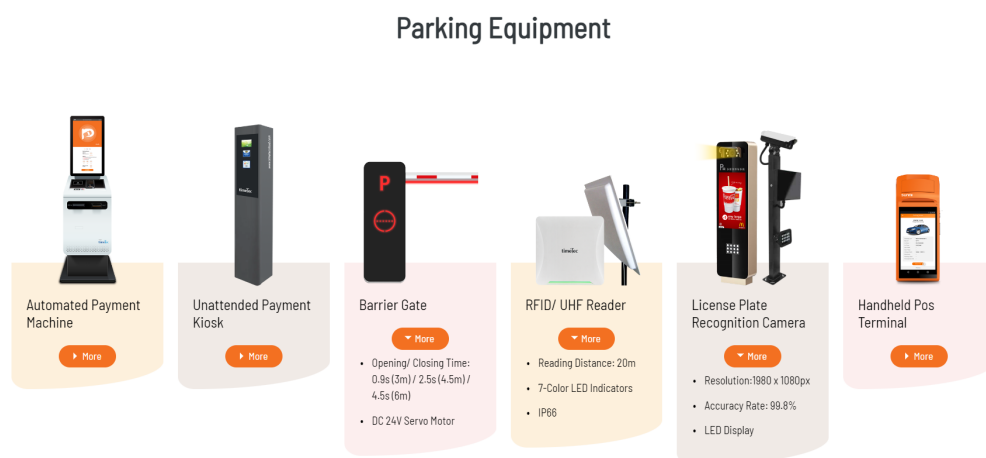


Figure2.1.2.3 Parking Equipment from timeTec

QR code technology has become a popular trend in parking system technology, particularly in residential settings where safe and efficient guest control is essential. Veemios has launched a visitor management system and community mobile application. Residents can use the system to pre-register visitors, which will generate a special QR

code that visitors will need to provide to guards when they arrive.[7] Security guards can scan the QR code to check the visitor's authorization, and admission is granted only if the QR code is valid. This technique improves security by limiting access to residential areas to pre-register visitors, lowering the danger of unauthorized entry. In addition, using QR codes simplifies the entry procedure, making it more convenient and faster than previous manual approaches.

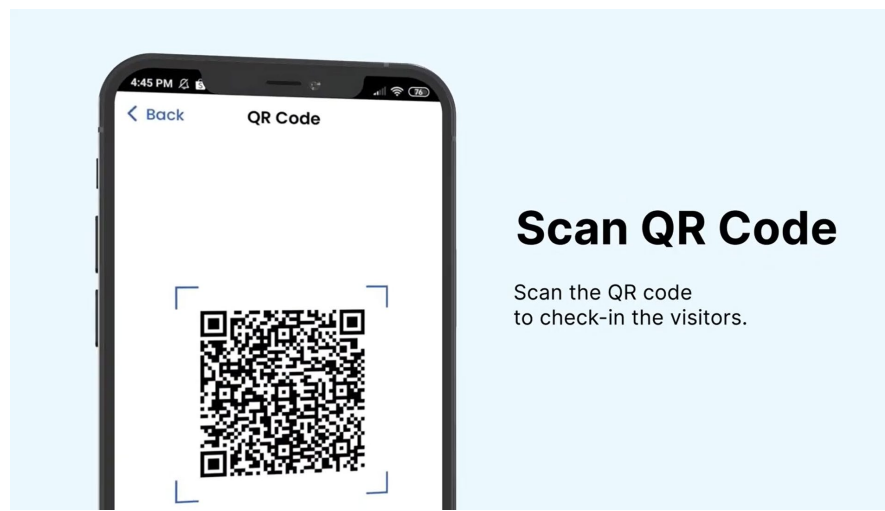


Figure2.1.2.4 QR code technology from Veemios

2.2 NFC-Based Authentication

2.2.1 Case Studies of NFC-Based Authentication

Authentication using near field communication (NFC) technology has become a prominent area of research, particularly in enhancing the security of access to personal and professional data

1. NFC-Based Authentication for Personal Computers [8]

With the increasing vulnerability of personal data to sophisticated hacking techniques, enhancing the security of authentication methods has become crucial. Wang and his team aimed to address this concern by developing an NFC-based authentication system specifically for Windows operating system logins. Their objective was to create a

solution that not only strengthens security but also simplifies the user authentication process.

The team implemented a two-factor authentication system that integrates NFC tags with traditional password entry. In this system, users are required to have an NFC tag attached to their smartphone. During login, the user taps their NFC tag on a reader connected to the computer, which serves as the first layer of authentication, followed by password entry. This dual-layer approach is designed to mitigate the risks associated with password-only systems.

The implementation of NFC technology in this context provided a significant improvement in both security and convenience. The study found that by requiring physical proximity of the NFC tag, the system effectively reduces the risk of unauthorized access. Additionally, the ease of use was enhanced as users could streamline the login process with a simple tap, reducing the need for repetitive password entry. The findings suggest that NFC technology has the potential to be widely adopted for secure logins in personal and professional environments. The dual-layer authentication model offers a significant security advantage without compromising on user convenience.

2. NFC-Based E-Ticket System [9]

The traditional e-ticketing systems, often relying on QR codes, can be cumbersome and pose security challenges. To address these issues, Li and his team explored the use of NFC technology to streamline and secure the e-ticketing process. The objective was to create a more efficient and secure method for ticket booking, storage, and verification.

The team developed an NFC-based e-ticket system involving three key stakeholders: platform managers, ticket service providers, and users. The system operates in two primary stages: OTA ticket booking and authentication. Users begin by downloading an e-ticket application from a digital store and purchasing tickets through the

application. The e-tickets are then sent via OTA to the users' NFC-enabled smartphones, where they are securely stored. For ticket verification, users simply wave their phones in front of a reader at the point of entry, eliminating the need to manually select and display a ticket within an application.

The NFC-based system was found to significantly enhance both security and user experience. The system's ability to store tickets securely on a user's phone and authenticate them with a simple tap offers a substantial improvement over QR-code-based systems, which require more steps for validation. This case study highlights the potential of NFC technology to revolutionize e-ticketing by improving efficiency and security.

Both NFC-based authentication systems demonstrate the technology's ability to enhance security and user convenience across different applications. Whether securing personal computer logins or streamlining e-ticketing processes, NFC offers a versatile and effective solution for modern authentication needs.

2.2.2 Comparison of NFC With Other Authentication Methods

The project will compare NFC with three other authentication methods Passwords, Biometrics, and QR Codes. The comparison will focus on various aspects such as security, convenience, cost, and versatility.

1. NFC

NFC provides strong security through proximity-based authentication. Compared to passwords and QR codes, the short distance lowers the chance of interception. To prevent unwanted access and data breaches, NFC usually transmits data via encryption. Thus, it is a secure option for private data and data transmission. NFC is highly convenient and fast as it requires only a tap or wave of a device. It is faster than QR codes that require scanning and manual entry [10]. The cost of NFC technology is moderate due to the need for NFC-enabled devices and readers. With the development

of technology, most mobile phones are equipped with NFC functions. Therefore, the cost is moderate compared to RFID and biometrics authentication. In terms of versatility, NFC is widely applicable in areas such as mobile payments, access control and authentication used.

2. QR codes

QR codes have limited security because the information is visual and can be easily copied. QR codes can be used maliciously to redirect users to harmful websites. QR codes are less convenient as they require the user to use a camera to capture the QR code. If the user's mobile phone does not have a QR code scanning function. They may need to install a QR code scanner application [11]. However, QR codes are less expensive. It can be implemented easily and at low cost. QR codes require a camera to scan, and the scanning process may be affected by lighting conditions and code quality. However, QR codes are highly versatile and can be used in many areas such as marketing, ticketing, and information sharing.

3. Biometrics

Since each person's physiological characteristics are unique and cannot be replicated, biometric technology provides a high level of security. Fingerprints, iris and facial recognition are different for everyone. Therefore, biometric authentication is very secure. Biometrics are so convenient that people can simply scan or touch a scanner. You can authenticate quickly. And users don't need to remember any passwords or carry any tools for authentication [12]. However, biometric identification requires specialized hardware and software for biometric identification, which undoubtedly requires high costs. Biometric authentication is highly versatile. People can use it to unlock their phones and clock in to work. It works in a safe environment.

4. Password

Passwords can offer high security if they are strong and complex. However, they are often vulnerable to attacks such as phishing, brute force, or social engineering. Passwords can be cumbersome to manage and remember. They require manual entry,

which can be time-consuming and prone to errors [13]. The cost for implementation is low but it is hard to manage. They are susceptible to theft and misuse and managing them can be challenging. However, the versatility for passwords is very high. It is universally applicable to most of the digital platforms and systems.

In conclusion, each authentication method offers distinct advantages and limitations. NFC provides robust security and convenience but comes with moderate costs. QR codes are inexpensive and versatile but have lower security and convenience. Biometrics offer high security and convenience but require significant investment in specialized hardware. Passwords, while low-cost and highly versatile, often face challenges with security and user management. The choice of authentication method depends on balancing these factors to meet specific needs and contexts effectively.

| Authentication Method | Security | Convenience | Cost | Versatility |
|-----------------------|------------------------------|-------------|----------|-------------|
| NFC | High | High | Moderate | Moderate |
| QR codes | Low | Moderate | Low | High |
| Biometrics | Very High | High | High | Moderate |
| Passwords | High (if strong and complex) | Low | Low | Very High |

Table 2.2.1 Summary of Comparison of NFC With Other Authentication Methods

2.3 Mobile Application Development

2.3.1 Case studies on Parking Systems using Mobile Application

In this project, two aspects of parking systems using mobile applications will be studied: residential and commercial. However, neither area typically employs independent parking management systems. In contrast, mobile applications for residential uses often integrate parking management into broader community management functions. Likewise, commercial parking management is often integrated into larger multi-

purpose applications. This section explores two applications, namely Veemios[7] and i-Neighbour [14] as case studies for residential environments and Sunway Super App for commercial parking management.

1. Veemios

Veemios is a mobile application designed for visitor management and community engagement. It is used by residents to manage visitor access and interact with management. While the application provides a range of community functions, its parking management capabilities are integrated into the broader visitor management system.

In Veemios, residents are required to pre-register their visitors. Once the registration is completed, Veemios generates a QR code that the resident shares with the visitor. Upon arrival at the residential area, the visitor presents the QR code to the security guard. The security guard then uses a QR scanner within the Veemios system to check in the visitor. Upon successful check-in, the resident is notified via the application that their visitor has been successfully logged.

One notable limitation of Veemios is its parking assignment process. After the visitor is checked in, the security guard is responsible for assigning a parking spot. The guard inputs the parking lot number into the system and provides the visitor with a paper slip indicating their assigned parking spot. However, the visitor still relies on the guard's verbal instructions to locate the specific spot, which can lead to confusion or errors, especially if the parking area is large or complex.

Veemios also includes an Issue Log feature, where residents can submit complaints or maintenance requests directly to management. This feature allows residents to track the status of their requests, providing transparency and ensuring that management takes timely action. Residents can view a history of their submitted issues, which helps them monitor whether their concerns are being addressed.

From the management's perspective, Veemios offers several tools to streamline operations. Management can access visitor traffic patterns through the application, which helps in organizing security workflows more efficiently. The visitor entry log, which includes detailed information provided by visitors during check-in, is also available to management. Additionally, Veemios allows management to monitor parking occupancy, providing them with a clear overview of visitor parking utilization and aiding in the effective allocation of parking spaces.

The Veemios application demonstrates how mobile technology can integrate parking management into a broader community management system. While it effectively manages visitor check-in and provides valuable tools for residents and management, the reliance on manual processes for parking assignment highlights areas for potential improvement. This case study illustrates the balance between technological integration and manual oversight in residential parking management systems.

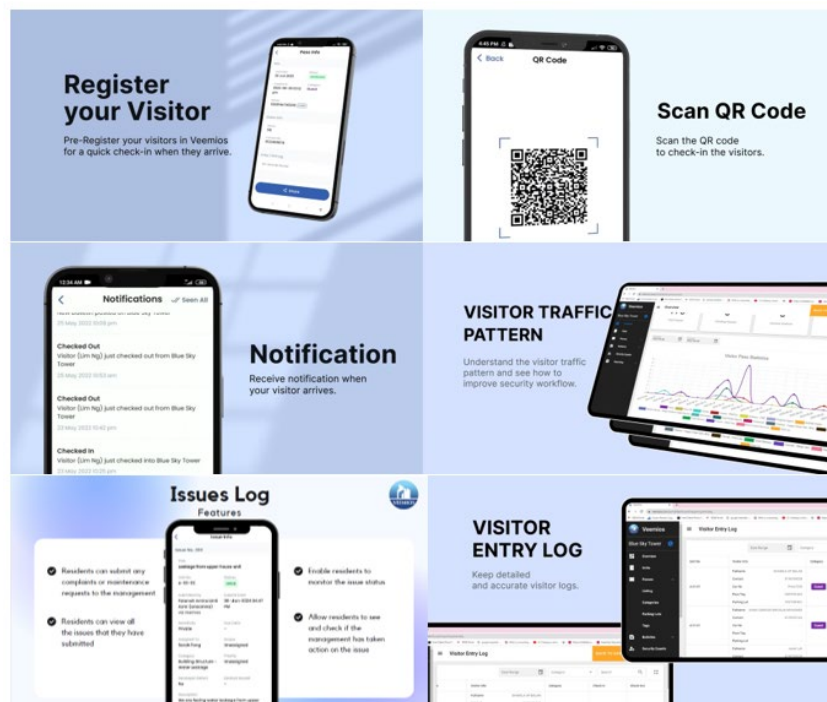


Figure 2.3.1.1 Parking Management System for Veemios

2. i-Neighbourhood

i-Neighbourhood is an application similar to Veemios, designed to streamline visitor management and enhance security within residential communities. This application allows residents to send invitations to visitors, who are then required to fill out an electronic form and upload a photo. Upon arrival at the residential area, visitors can either scan their face or use a QR code to open the barrier gate. Residents receive notifications when their visitors check in or out, making the process convenient and ensuring that only approved visitors gain access to the premises, thereby significantly improving security.

One of the standout features of i-Neighbourhood is its blacklist function. This feature allows the community to blacklist suspicious visitors, preventing them from entering the residential area. This is an excellent tool for maintaining community safety and harmony, as it ensures that only trustworthy individuals can enter.

In terms of parking management, i-Neighbourhood employs advanced technologies like automatic check-in/out barrier gates and Optical Character Recognition (OCR). The automatic barrier gate only opens for registered visitors, and as soon as the gate opens, a check-in or check-out notification is sent to the resident. The use of OCR technology further enhances the visitor registration process by automatically filling in visitor details from a photograph of their identification ID. This not only saves time but also makes the entire process more efficient and user-friendly.

Overall, i-Neighbourhood integrates several advanced features to improve both security and convenience within residential communities, making it a highly effective tool for modern visitor and parking management.

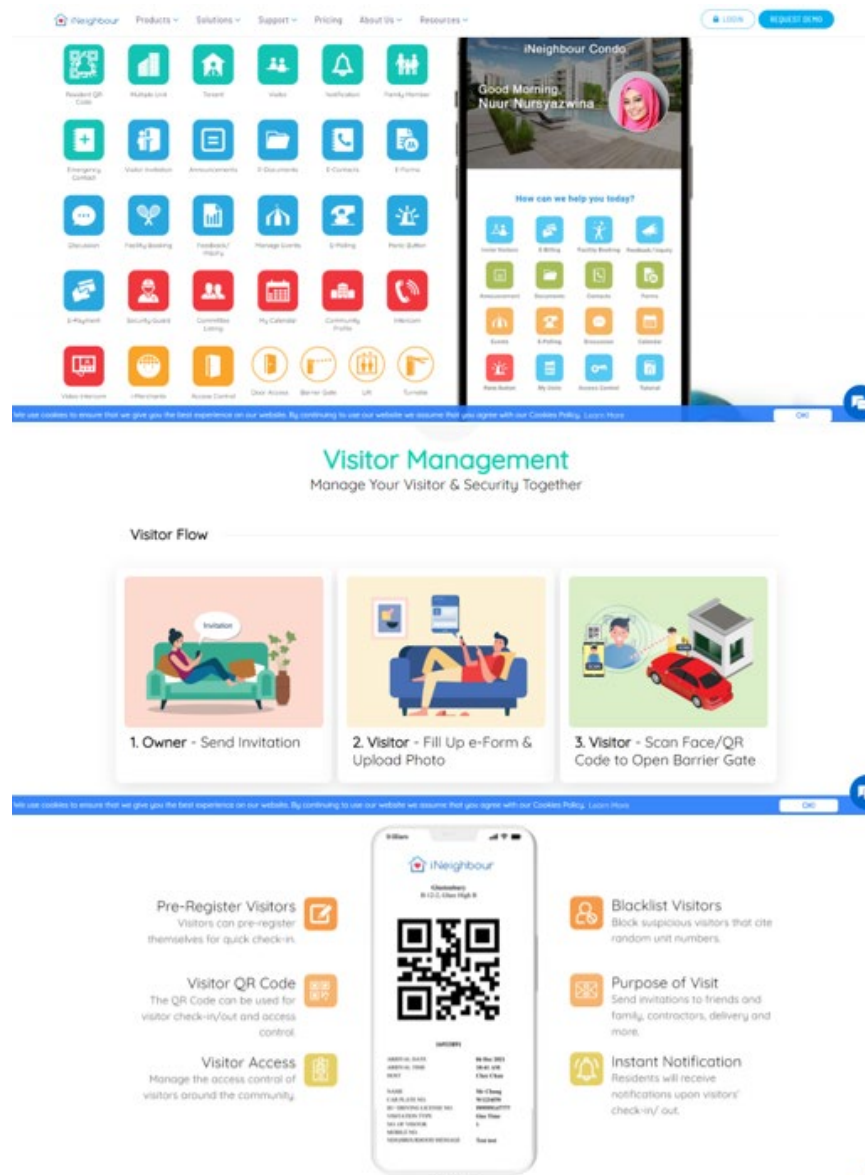


Figure 2.3.1.2 Parking Management System for i-Neighbour

3. Sunway Super App

The Sunway Super App is a comprehensive application offered by Sunway, designed to enhance the customer experience across its properties. One of the key components of this application is the Sunway Smart Parking system [15], a commercial smart parking management solution implemented across several Sunway properties, including Sunway Pyramid, Sunway Pinnacle, Sunway University, Sunway Medical Centre, Sunway Geo Avenue, and Sunway Clio Hotel.

Sunway Smart Parking leverages Automated License Plate Recognition (ALPR) technology to automatically recognize vehicle license plates, enabling seamless entry and exit without the need for physical tickets. This system not only streamlines the parking experience but also reduces the chances of parking tickets being lost or misplaced, offering a hassle-free experience for users.

In addition to the ALPR, Sunway Smart Parking supports cashless payments. Visitors can conveniently pay for their parking fees through the Sunway Super App, eliminating the need to carry cash or queue at payment kiosks. This feature is particularly useful in today's digital age, where many people have become accustomed to contactless and cashless transactions.

Sunway Smart Parking exemplifies how technology can be integrated into commercial parking management to offer a more efficient, user-friendly experience. By utilizing advanced technologies like ALPR, Sunway has successfully created a parking solution that meets the needs of its visitors while also aligning with modern expectations for convenience and ease of use.

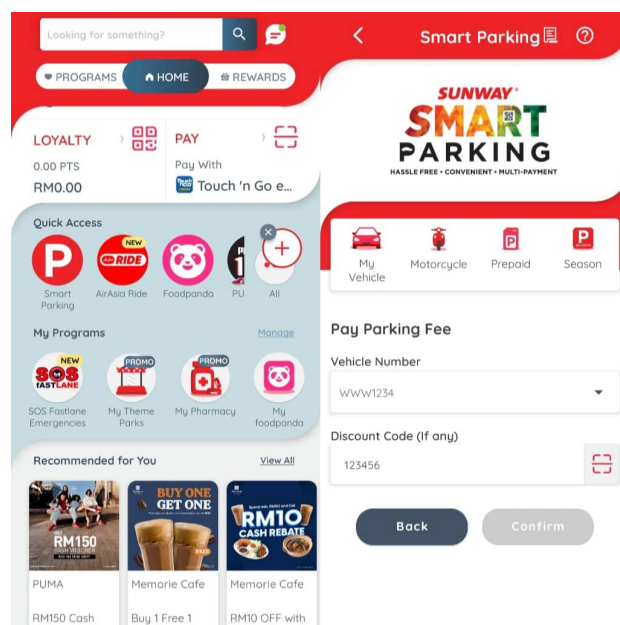


Figure 2.3.1.3 Sunway Smart Parking

2.3.2 Importance of user-friendly in Mobile Application Development

User-friendliness plays a vital role in mobile application development and determines the way users interact with and perceive the application. A well-structured, intuitive interface can greatly enhance the overall user experience, making it easier for users to navigate the application's features. This positive interaction leads to higher satisfaction, as users are more likely to prefer applications that are simple, beautiful, and meet their needs [17].

Enhanced user experience is at the core of a successful mobile application. User-friendly applications ensure tasks are easy to perform, information is easy to find, and operations are clear and concise. Features like logical layout, consistent design elements, and fast response times all contribute to a smoother experience. When users can achieve their goals quickly without feeling confused or frustrated, they are more likely to find the application valuable.

In addition to improving user experience, user-friendly mobile applications also have a direct impact on user retention. Users are less likely to abandon an application they find easy and enjoyable to use. In a competitive market, users have many choices, so it's important to keep them engaged. Applications that are difficult to navigate or require too much effort to use quickly turn users toward more accessible alternatives. Conversely, a user-friendly application provides a smooth, hassle-free experience that encourages users to return again and again.

Additionally, user-friendliness can lead to higher satisfaction. Applications with well-designed interfaces make it easier for users to take the desired action. For example, a streamlined checkout process can reduce abandonment rates, while a clear call-to-action button can guide users toward completing a transaction. Higher satisfaction is achieved when applications make these processes simple and intuitive.

In summary, user-friendliness is crucial in mobile application development as it enhances user experience, improves retention, and increases satisfaction. Developers who prioritize intuitive design, clear navigation, and overall positive user interaction will create applications that not only meet user needs but also stand out in the market.

2.4 Proposed Residential Parking Management System with Mobile NFC Authentication

The proposed parking management system is specifically designed for residential use and integrates Near Field Communication (NFC) technology for secure vehicle authentication. A key feature of this system is the use of NFC stickers, which store essential vehicle and resident data for verification purposes. The NFC tag stores information such as the car plate number, vehicle model, color, and assigned parking lot. This allows management to verify vehicle identity simply by tapping the NFC tag—eliminating the need to search through paper records to confirm whether a parked vehicle matches its registered details.

This NFC-based verification process significantly enhances both efficiency and security. If the data retrieved from the NFC tag does not match the physical vehicle, it may indicate unauthorized usage or sticker transfer. If no data can be retrieved at all, it may suggest that a counterfeit sticker is being used. Compared to traditional handwritten methods, this approach is more secure, reliable, and less prone to human error.

To authorize visitor entry, only residents are permitted to register their visitors through the application. Upon successful registration, the system generates a unique reference code, which the resident shares with the visitor. The visitor then inputs this code into the app upon arrival to confirm their identity and receive an assigned parking spot. This ensures that only authorized guests can access the premises, maintaining a high level of control and security. Visitors also have access to a view map feature that guides them visually to their assigned parking space.

In addition, the application provides a reporting feature that empowers residents to report incidents of illegal parking or unauthorized vehicles. These reports are visible to management in real time, enabling quick responses to violations. To assist staff with oversight, the system includes a management floor plan view that displays the current availability of visitor parking lots, helping to monitor and allocate spaces efficiently.

The integration of NFC authentication differentiated floor plans for residents and visitors, real-time reporting, and centralized management tools results in a secure, efficient, and user-friendly residential parking management system tailored for modern communities.

| Feature | Veemios | i-Neighbour | Sunway Super App | Proposed NFC Parking System |
|-------------------------------------|-----------------------------------|--------------------------------------|----------------------------------|---|
| Platform Type | Residential | Residential | Commercial | Residential |
| Visitor Access Method | QR code shared by resident | QR code or face recognition | ALPR (License Plate Recognition) | Reference code shared by resident |
| Visitor Registered By | Resident | Resident | Not required (open to public) | Resident |
| Parking Spot Assignment | Manually by security guard | Automatically through barrier system | Automatically via ALPR | Auto-assigned by system based on reference code |
| Vehicle Authentication | Manual by guard | OCR + Barrier Automation | ALPR | NFC tag scanned to match vehicle details |
| Fake Sticker Detection | Not available | Not available | Not available | Yes – detects mismatch or unreadable tags |
| Real-Time Monitoring for Management | Basic logs (visitor check-in/out) | Entry/exit logs | Parking occupancy | Live overview of visitor parking availability |

| | | | | |
|------------------------------|------------------------|------------------------------|----------------------------|---|
| User Reporting Function | General issue log | Not explicitly stated | Not available | Illegal parking report with real-time updates |
| Navigation/Floor Plan | Not provided | Not provided | Not applicable | View Map Feature for Visitor |
| Cashless Payment Support | Not applicable | Not applicable | Yes | Not applicable |
| Security Enhancement | Basic visitor check-in | Blacklist + ID verification | ALPR access control | NFC verification |
| Technology Integration Level | Medium – QR & logging | High – OCR, face recognition | Very High – ALPR, cashless | High – NFC, smart map, reporting, auto assignment |

Table 2.4.1 Comparison Table of the Proposed Residential Parking Management System and Reviewed Parking Management Systems

CHAPTER 3

SYSTEM METHODOLOGY

3.1 System Development Models

System Development Models are structured approaches or methodologies used in the process of designing, developing, and deploying software systems. These models provide a step-by-step process for planning, creating, testing, and maintaining software, ensuring the product meets user requirements and maintains high quality. Different models cater to various project needs, allowing flexibility or rigidity depending on the complexity and nature of the system.

In this project, three system development models are being compared. The three system development model are Waterfall, Agile, and Prototype.

3.1.1 System Development Model 1: Waterfall

The Waterfall model is a traditional and linear approach to software development. It is structured in a sequence of phases where each phase must be completed before the next one begins. This model is best suited for projects that requirements are well-defined from the start and are unlikely to change throughout the development process.[18]

The waterfall model follows a rigid phase-by-phase schedule, each step must be finished completely before proceeding to the next. Unlike Agile model, where development and testing may occur concurrently, Waterfall model sees each step as separate. A phase can only start when the preceding one has finished. Throughout the process, waterfall model places an emphasis on detailed documentation. Each phase creates detailed reports and records to ensure that the entire process is well-defined. Changes are difficult for waterfall to handle; once a project is underway, it is challenging to accommodate new requirements.

For development of a parking management system, the Waterfall model might be appropriate if the system requirements are clear from the beginning and not expected to change.

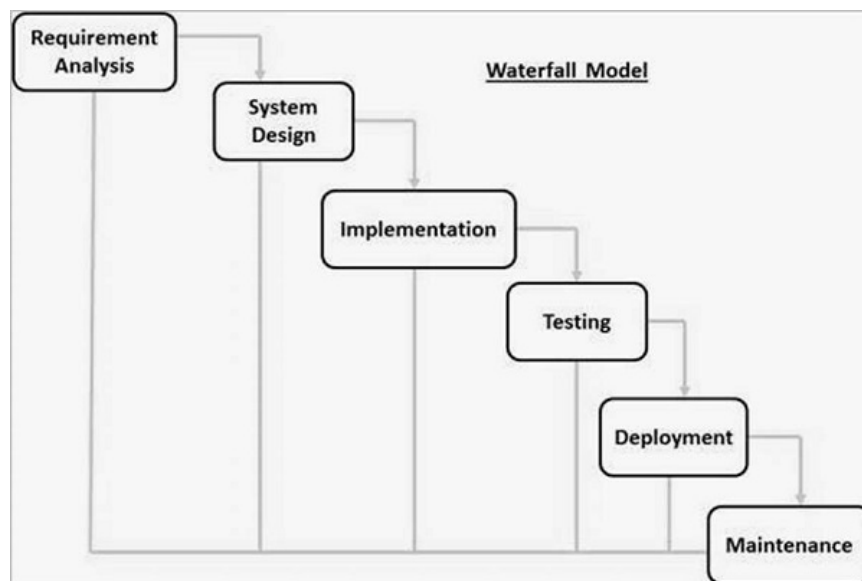


Figure 3.1.1 Waterfall Model

3.1.2 System Development Model 2: Agile

Agile model is a popular system development methodology, especially for projects requiring flexibility, continuous improvement, and client collaboration. Agile model emphasizes iterative development, in which requirements and solutions emerge from the collaborative efforts of cross-functional teams. The methodology emphasizes adaptability and the delivery of functional software in short cycles known as sprints, which allows for rapid reassessment and adjustments during the development process.[19]

Agile model is a systematic development method for iterative and incremental development. It divides the project into small and manageable sprints. Each sprint delivers an increment of work product that can be tested and evaluated. Agile model involves customer communication, and customer feedback can improve product development. Agile model is a flexible system development method. It allows changes to be made to the project without significantly affecting the overall schedule.

For the development of parking management system, Agile model enables the system the system developed and deployed in phases. Initial of the project, sprints can focus on the core features such as report illegal parking, visitor registration and verification of NFC sticker. Then can continue with the advanced feature such as real-time parking analytics. Agile model is flexibility allows for quick adaptation to new stakeholder needs, regulatory changes, or evolving new feature without disrupting the project

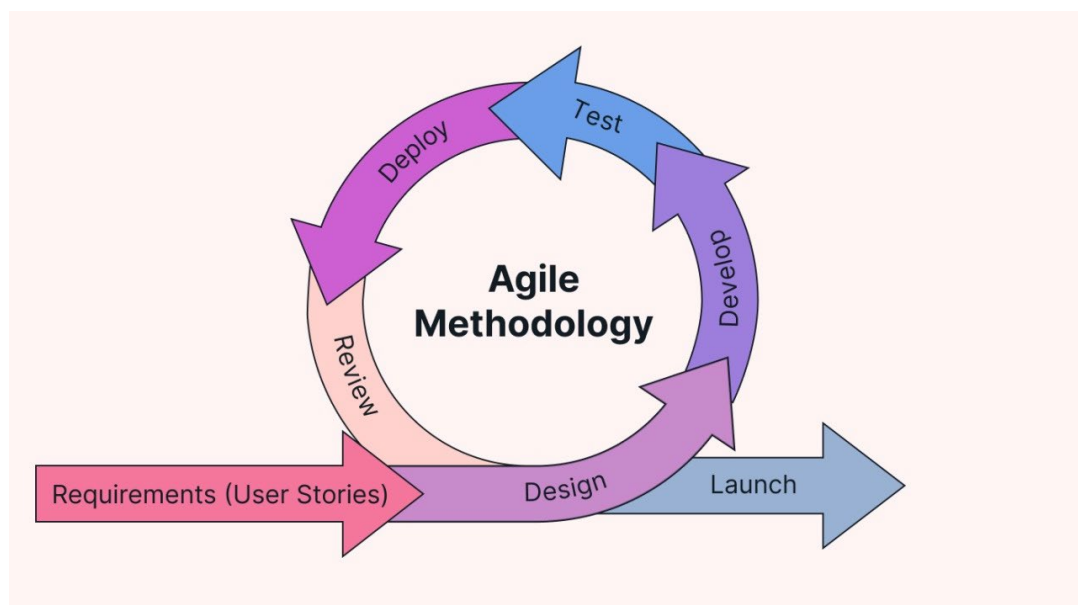


Figure 3.1.2 Agile Model

3.1.3 System Development Model 3: Prototype

The Prototyping Model is a development approach that focuses on creating a working prototype of the system early in the development process [20]. The prototype is an early version of the software that simulates its main features, allowing stakeholders to interact with it and provide feedback. The model emphasizes understanding the user requirements better by building a working, though incomplete, version of the software.

The goal is to develop an early, functional prototype of the system to help visualize how the final product will work. This helps clarify requirements that may not be initially well-defined. users are heavily involved throughout the development process. They

provide continuous feedback based on their interactions with the prototype. The prototyping model is useful in for project where the project requirements are not fully understood at the beginning. The prototype serves as a tool to gather and refine requirements as the project progresses.

The prototyping model is well-suited for the development of a parking management system, especially when the requirements are unclear or subject to change. In developing a parking management system, initial prototypes could focus on core features like report illegal parking or visitor registration. Users can interact with the system early on to provide feedback. The Prototyping model is ideal for parking systems that might need to adapt to changing technologies or user demands.

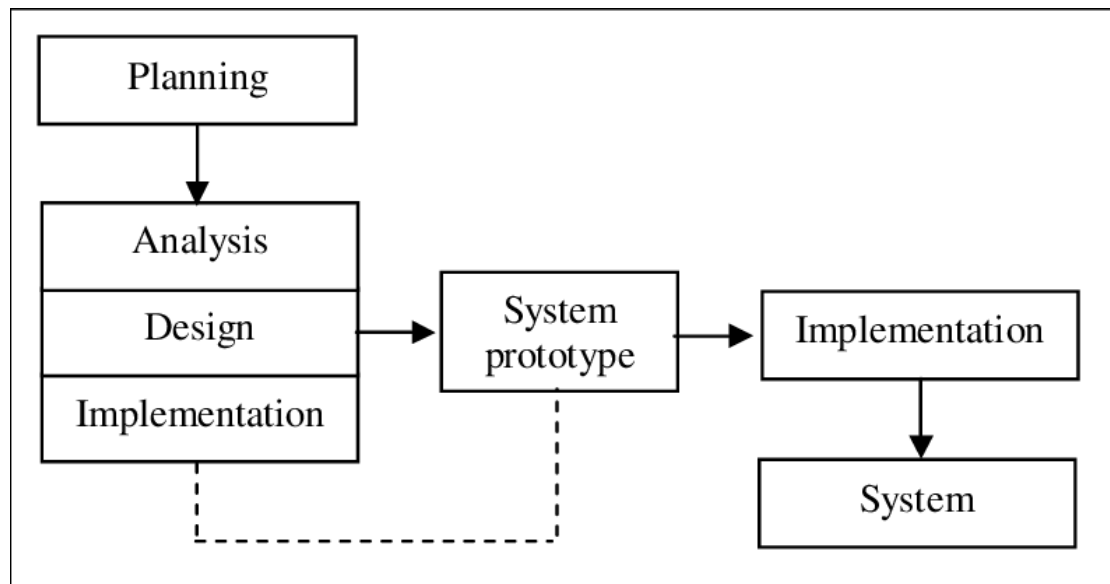


Figure 3.1.3 Prototype Model

3.1.4 Selected Model: Prototype Model

The methodology chosen for this project is the Prototype Model [14] , which comprises six distinct steps, each contributing to the systematic development of the system. These steps are designed to enhance the efficiency of the development process and ensure that potential issues are identified and addressed early on.

Step 1: Planning

In the planning phase, system information is gathered by researching similar projects to understand potential challenges. The project scope, objectives, and problem statement are clearly defined to establish the direction of the project. This phase identifying what the project aims to achieve and the core issues it intends to address.

Step 2: Analysis

During the analysis phase, a literature review was conducted on three NFC systems and three parking management systems to identify the strength and weakness. By analyzing these existing applications, the project gathered detailed requirements based on insights from the review to form the design of the project.

Step 3: Design

The design phase focuses on defining the methodology and overall design of the mobile application. For this project, the methodology is prototyping methodology. Based on the project scope, objectives, and problem statement, use case diagram and block diagrams are created to outline the system's structure and functionality. This phase ensures that the application's design is aligned with the project's goals and that all necessary components are properly planned and integrated.

Step 4: Implementation

During the implementation phase, the application is developed using Android Studio and Firebase based on the use cases and block diagrams created during the design phase. This involves coding the functionality and user interface. The functionality implementation is mainly coded in the Java language. At the same time, Firebase provide authentication and data storage, and ensure the application runs according to the intended design.

Step 5: System Prototype

In the system prototyping phase, a model of the application is developed to demonstrate its features and functionality. The prototype allows for early testing and validation of the system's design and usability. Feedback from users is gathered to identify any issues or areas for improvement. The prototype serves as the basis for further development, ensuring that the final product meets user expectations and requirements. If user evaluation is satisfactory, the implementation of the complete system will proceed. Otherwise, the design and system prototype will be refined until the user is satisfied.

Step 6: System Deployment

Once the prototype is refined and verified to meet user expectations, the entire system is ready for deployment. This includes finalizing the code, conducting thorough testing to ensure stability and performance, and then releasing the application to production.

3.2 System Requirements

3.2.1 Hardware Components

The hardware designated for this project encompasses smartphones utilizing the Android operating system, serving a dual purpose for performance testing and functioning as NFC readers for testing NFC authentication. Additionally, NFC tags will be utilized for authentication testing scenarios. Furthermore, laptops will be employed for application programming tasks.

| | | |
|---------------------------|----------------------|---|
| Smartphone Specification: | Processor: | MediaTek Dimensity 7050 |
| | Operating System: | Android 14, ColorOS version V14.0 |
| | RAM: | 12 GB |
| | Manufacturer: | OPPO |
| | NFC: | Yes |
| Laptop Specification: | Processor: | Intel® Core (TM) i7-10750H CPU @ 2.60GHz, 2592Mhz |
| | Operating System: | Microsoft Windows 11 Home Single Language |
| | System Type: | x64-based PC |
| | RAM: | 8GB |
| | Manufacturer: | Micro-Star International Co.,Ltd. |
| NFC Tags | NFC Forum Type: | NFC Forum Type 2 Tag |
| | Chip Type: | NTAG213, 215, 216 |
| | Operating frequency: | 13.56 MHz |
| | Memory: | 540Bytes |
| | Usable Memory: | 504Bytes |
| | Protocol: | ISO/IEC 14443 A |
| | Reading distance: | 0-10cm |

Table 3.2.1.1 Smartphone, NFC Tags and Laptop Specification

The smartphone is used as the primary NFC tool in this project. It serves both as an NFC reader (to verify tag data) and an encoder (to write car details onto NFC tags) using Android's built-in NFC functionality. This eliminates the need for a dedicated external USB NFC reader. This laptop is used for software development, testing, and building the Android application using Android Studio. These tags are used for storing vehicle and resident information, which can be read and written using the NFC-enabled smartphone.

3.2.2 Software Components

In this project, a suite of software tools has been carefully selected to facilitate efficient and effective development. The tools include Android Studio, Firebase, each serving a specific purpose in the development process.



Figure 3.2.2.1 Android Studio

Android Studio:

Android Studio is a development tool specifically for creating Android applications. In this project, Android Studio is the primary tool used to develop the parking management system application using the Java programming language.

Minimum Requirements to Install Android Studio:

| Requirement | Specification |
|----------------------|---|
| Operating System | Windows 10/11 (64-bit), macOS, or Ubuntu/Linux (64-bit) |
| RAM | Minimum 8 GB RAM (16 GB recommended) |
| Disk Space | At least 8 GB of free disk space |
| Java Development Kit | Bundled with Android Studio |
| Screen Resolution | Minimum 1280 x 800 |

Table 3.2.2.1 Minimum Requirements to Install Android Studio



Figure 3.2.2.2 Firebase

Firebase:

Firebase, supported by Google, is our go-to platform for creating and developing applications. In this project, Firebase serves as the main database, storing input data, information, and handling role authentication.

Minimum Requirements to Use Firebase (Web Access):

| Requirements | Specification |
|---------------------|--|
| Internet Connection | Stable internet access is required |
| Web Browser | Latest version of Chrome, Firefox, Safari, or Edge |
| Google Account | Required to access Firebase Console |

Table 3.2.2.2 Minimum Requirements to Use Firebase

3.3 Functional Requirements

The functional requirements in this report are divided into four sections, which are User Registration and Authentication, Resident Functionalities, Management Functionalities, and Visitor Functionalities. These sections outline the essential tasks and features that each type of user must be able to perform within the system. Also, each section details the specific functionalities required to ensure that users can interact with the system effectively according to their respective roles

1. User Registration and Authentication

a. Register an account

User must be able to create an account by providing an email address and password.

CHAPTER 3

b. Login

User must be able to login using their email address and password.

c. Forgot Password

User should be able to reset their password if they forget it.

d. Role assign

The system must assign roles such as resident and management based on the user's account type

2. Resident Functionalities

a. Report Illegal parking

Resident must be able to report illegal parking and the system must be able to store the report to the firebase.

b. View NFC Parking Sticker Details

Residents should be able to view the details of their NFC parking stickers using the system.

c. Reserve Visitor Parking

Residents must be able to reserve parking spots for visitors by entering the date and time, and then receive a reference code to share with the visitor.

d. View Illegal Parking Report History

Residents should be able to view a history of all the illegal parking reports they have submitted.

3. Management Functionalities

a. Receive Illegal Parking Reports

Management must be able to receive and view illegal parking reports submitted by residents. The report must receive real-time so that the management can take necessary action immediately for the illegal car.

b. Manage Visitor Parking Status

Management should be able to update and manage the status of visitor parking spots by uploading a csv file. The table of visitor parking status must provide the latest status.

c. Write NFC Parking Stickers Details

Management must be able to write vehicle-related information (such as car plate, color, model, and assigned lot) to NFC stickers using their mobile device. This is done through the app's interface, ensuring that each NFC tag contains correct and traceable data.

d. Update NFC Parking Sticker Details

Management should be able to update the information on NFC parking stickers, such as correcting errors or changing details.

e. View NFC Parking Stickers

Management must be able to view the details of NFC parking stickers for verification purposes.

4. Visitor Functionalities

a. Anonymous Access

Visitors should be able to access the system without needing to create an account, using anonymous authentication.

b. Enter Visitor Details

Visitors must be able to enter their personal details, car plate number and reference code into the system. The reference code provided by resident to verify their reservation.

c. View Assigned Parking Spot

After entering their details and reference code, visitors should be able to view their assigned parking spot and a floor plan.

3.4 Project Milestones

3.4.1 Project 1 Timeline

| Activity | Period | | | | | | | | | | | |
|---|--------|----|----|----|----|----|----|----|----|-----|-----|-----|
| | W1 | W2 | W3 | W4 | W5 | W6 | W7 | W8 | W9 | W10 | W11 | W12 |
| 1. Project Analysis | | | | | | | | | | | | |
| Identify motivation and problem statement, project scope, and project objective | | | | | | | | | | | | |
| Conduct research for the similar system | | | | | | | | | | | | |
| 2. Project Design | | | | | | | | | | | | |
| System design | | | | | | | | | | | | |
| Design user interface for the application | | | | | | | | | | | | |
| 3. Project Implementation and Development | | | | | | | | | | | | |
| Set up development Environment | | | | | | | | | | | | |
| Develop User Interface | | | | | | | | | | | | |
| Implement Core Feature | | | | | | | | | | | | |
| Implement Firebase Intergration | | | | | | | | | | | | |
| 4. Testing and Improve | | | | | | | | | | | | |
| Testing and improve core feature | | | | | | | | | | | | |
| Improve user interface | | | | | | | | | | | | |

Figure 3.4.1.1 Gantt Chart of Project 1 Timeline

The timeline for Project 1 begins with the Project Analysis phase in Week 1. During this week, the project's foundation is established by defining the motivation behind the project, articulating the problem statement, and outlining the contributions it will make. Additionally, the project scope is set to clarify the boundaries of the work, and specific objectives are defined to guide the project's development.

In Week 2, the focus shifts to a Literature Review. This review is divided into several key areas: authentication in parking systems, NFC-based authentication, and mobile application development. It covers traditional parking methods like paperwork and guard patrols, examines current trends in parking technologies, and explores case studies on NFC-based systems. The review also includes a comparison of NFC with other authentication methods and discusses the importance of a user-friendly interface in mobile applications, supported by relevant case studies.

The project then moves into the Project Design phase, spanning Weeks 3 and 4. In Week 3, a suitable system design is identified, which includes determining the architecture and components necessary for the project. Week 4 is dedicated to refining this system design and creating the user interface for the application, ensuring that it is both functional and user-friendly.

From Weeks 5 to 10, the Project Implementation and Development phase takes place. Week 5 is focused on setting up and configuring the software environment. In Week 6, a prototype of the project is created, and the initial user interface design is developed. The period from Week 6 to Week 10 is then used for integrating Firebase and implementing core features such as login, sign-in, illegal parking reporting, and visitor parking management.

The final phase, Testing and Improvement, occurs from Weeks 9 to 12. This phase is dedicated to testing the core features of the project, identifying any issues, and making necessary improvements. Additionally, the user interface is improved to ensure it is easy to use and user-friendly.

3.4.2 Project 2 Timeline

| Activity | Period | | | | | | | | | | | |
|---------------------------------------|--------|----|----|----|----|----|----|----|----|-----|-----|-----|
| | W1 | W2 | W3 | W4 | W5 | W6 | W7 | W8 | W9 | W10 | W11 | W12 |
| 1. Project Revising | | | | | | | | | | | | |
| Review the progress of FYP1 | | | | | | | | | | | | |
| 2. Project Design | | | | | | | | | | | | |
| Design NFC Authentication Integration | | | | | | | | | | | | |
| 3. Project Development | | | | | | | | | | | | |
| Software/ Hardware Setup | | | | | | | | | | | | |
| NFC Integration | | | | | | | | | | | | |
| Develop Core Features | | | | | | | | | | | | |
| Software Completion >50% | | | | | | | | | | | | |
| Software Completion >70% | | | | | | | | | | | | |
| Software Completion >100% | | | | | | | | | | | | |
| 3. Testing and Improve | | | | | | | | | | | | |
| System Performance Evaluation | | | | | | | | | | | | |
| System Testing Setup and Result | | | | | | | | | | | | |
| Perform Unit Testing | | | | | | | | | | | | |
| User Acceptance Testing | | | | | | | | | | | | |

Figure 3.4.2.1 Gantt Chart of Project 2 Timeline

The timeline for the final year project is a well-structured plan that spans over 12 weeks, guiding the development process through several key phases such as project revising, project design, project development, and testing & improvement.

The project begins in the first two weeks with a thorough review of the progress made in the initial phase of the Final Year Project (FYP1). This initial phase is crucial as it allows assess what has been achieved so far. The review process ensures a clear understanding of the project's status and sets the stage for the work that follows.

The design phase follows in weeks two to three, focusing on the crucial task of integrating NFC authentication into the system. This phase involves the conceptual and technical design of the NFC features, which are central to the project's goals. The design work during this period is essential for ensuring that the NFC integration will be both functional and aligned with the overall project objectives.

As the development phase progresses, core features of the software are built out between weeks six and nine. During this period, the project reaches significant milestones, with the software completion reaching 50% by week seven and 70% by

week eight. This incremental progress ensures that the development is on track, with each milestone representing a major achievement in the project's lifecycle. By the end of week ten, the core features are fully developed, marking the point where the software is 100% complete.

In the final weeks, the focus shifts to testing and improvement. System performance is evaluated in week ten, followed by thorough system and unit testing in weeks eleven and twelve. The project concludes with user acceptance testing in week twelve, ensuring that the system meets both technical standards and user expectations, readying it for deployment.

3.5 Estimated Cost

There is no significant cost associated with the software utilized for the development of this Parking Management System with Mobile NFC Authentication. All software tools employed in the project are freely available and can be downloaded from the internet without incurring any charges. These include:

- **Android Studio** – an official integrated development environment (IDE) for Android application development provided by Google at no cost.
- **Firebase** – a cloud-based platform offered by Google that provides services such as authentication, real-time database, and cloud storage. All features used in this project are accessible under Firebase's free tier, which is sufficient for the project scope.

The development of this system was carried out using a personally owned laptop and NFC-enabled smartphone, so no additional hardware purchase was necessary for coding, debugging, or testing the mobile application itself. This significantly reduced development costs.

The only expenditure incurred pertains to the procurement of NFC (Near Field Communication) tags required for testing the system's NFC functionality. Three types of NFC tags were purchased in equal quantities to evaluate compatibility and performance. The cost breakdown is shown below:

| NFC Tag Type | Unit Price (RM) | Quantity | Total Cost (RM) |
|----------------------|-----------------|----------|-----------------|
| NTAG213 | 0.91 | 3 | 2.73 |
| NTAG215 | 1.15 | 3 | 3.45 |
| NTAG216 | 1.35 | 3 | 4.05 |
| Total Estimated Cost | | | 10.23 |

Table 3.5.1 Estimated Cost Calculation

Therefore, the total estimated cost for this project amounts to **RM10.23**, which exclusively covers the hardware cost associated with NFC tag acquisition. No additional cost has been incurred for software or development tools.

3.6 Concluding Remark

In conclusion, Chapter 3 provided a comprehensive overview of the system development methodology adopted for the creation of the Parking Management System. This chapter began by exploring and evaluating three prominent system development models: Waterfall, Agile, and Prototype. Each model was analysed in terms of its process structure, advantages, disadvantages, and relevance to the current project. While the Waterfall model provides a structured and sequential approach, and the Agile model emphasizes iterative cycles and flexibility, it was the Prototype model that emerged as the most suitable methodology for this project.

The Prototype model was selected due to its iterative nature and its ability to incorporate continuous user feedback throughout the development process. This model is particularly beneficial for projects with evolving requirements or where the users are unsure of the full scope at the beginning. By using prototypes, the development team is able to present early versions of the system to stakeholders, collect their feedback, and refine the system iteratively until it meets the desired standards.

The chapter then detailed the six phases of the Prototype model, namely Planning, Analysis, Design, Implementation, System Prototyping, and Deployment. Each phase was explained in terms of its objectives and contributions to the overall system development. For instance, the planning and analysis phases allowed the team to gather requirements and study existing systems, while the design and implementation phases focused on creating use case diagrams, coding the system using Android Studio, and integrating Firebase services for authentication and data storage. The system prototyping phase enabled user interaction and feedback, which guided further refinement of the application. Finally, the deployment phase ensures the readiness of the system for real-world usage after thorough testing.

Following the methodology section, the chapter elaborated on the system requirements, including both hardware and software components necessary for development and testing. The specifications of smartphones, laptops, NFC tags, and

NFC readers were listed to illustrate the technological environment used for building and running the application. In terms of software, Android Studio and Firebase were chosen for development and backend services, respectively.

The functional requirements were also discussed in detail, categorized based on user roles: Residents, Management, and Visitors. Each role was assigned a set of specific features aligned with the overall goals of the Parking Management System. These functionalities ensure a seamless user experience, from user registration and NFC sticker management to illegal parking report submission and visitor parking management.

Overall, this chapter establishes a solid foundation for the system's development by presenting a well-justified choice of methodology and providing a clear roadmap for system construction. The combination of a structured prototyping approach, well-defined hardware/software requirements, and a detailed listing of system functionalities ensures that the system is developed in a user-centred, efficient, and effective manner. This chapter sets the stage for the implementation and testing phases discussed in the following chapters, ensuring the project remains on track and aligned with its objectives

CHAPTER 4

System Design

4.1 System Architecture

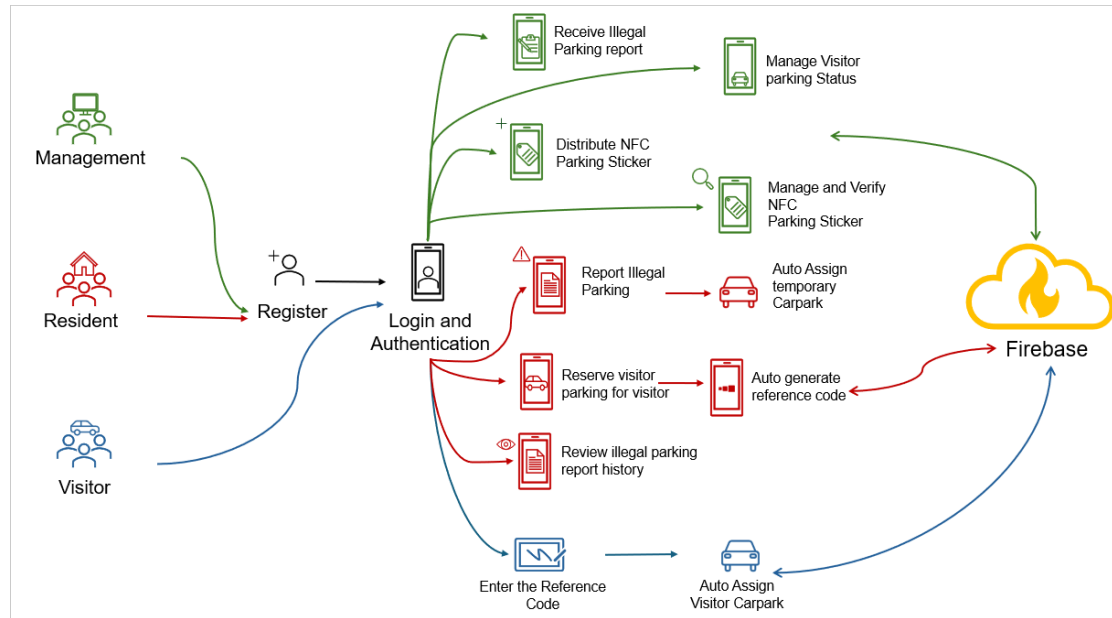


Figure 4.1.1 Block Diagram

The system design outlined in this report details a sophisticated residential parking management application tailored to serve three key user groups: Management, Residents, and Visitors. The architecture is crafted to ensure secure, efficient, and user-friendly management of parking spaces within a residential community, leveraging modern technologies such as Near Field Communication (NFC) and Firebase to enhance functionality and data security.

At the core of the system is the Login and Authentication module, which serves as the initial point of access for all users. Integrated with Firebase Authentication, this module ensures that only authorized individuals can interact with the system, and that all data exchanges are securely handled.

For residents, the system provides a comprehensive set of tools to manage their parking responsibilities. Residents can report illegal parking incidents directly through the app, which are submitted in real time to the management team for immediate action. They can also view their NFC Parking Stickers, which are assigned to their vehicles. These NFC tags store critical vehicle data, allowing management to tap and verify sticker authenticity—eliminating the risk of forgery commonly found in handwritten paper systems. Another key feature is the ability for residents to register visitors through the app. After registration, the system generates a reference code, which the resident shares with the visitor. This process ensures that only authorized guests can access the premises and helps the management maintain full control of parking space allocation.

Visitors do not register themselves in the system. Instead, they receive a reference code from a resident, which they must enter into the application upon arrival. Once verified, the system automatically assigns an available visitor parking space. The app also provides visitors with a View Map feature that includes a step-by-step navigation guide to their assigned parking lot, ensuring they can easily locate their designated space within the community.

The management team has access to a robust suite of features for monitoring and managing the parking facilities. They can receive and respond to illegal parking reports, verify NFC parking stickers by scanning the tags, and ensure that the registered vehicle matches the tag's stored data. This enhances the integrity of the parking system and prevents sticker misuse. Additionally, management can update NFC data, manage visitor parking statuses, and monitor overall occupancy. Parking spot data can be managed via CSV uploads, and visitor parking availability is displayed in real time for oversight.

The system's backend is powered by Firebase, which handles secure data storage and real-time synchronization across all features. This is particularly important for time-sensitive processes such as reporting violations or assigning parking to visitors.

Firestore ensures that sensitive user data, authentication credentials, vehicle details, and NFC tag content are protected from unauthorized access.

In conclusion, the redesigned system addresses the needs of Management, Residents, and Visitors by providing secure, efficient, and user-centered parking features. The resident-initiated visitor registration flow ensures control and accountability, while NFC-based verification and guided navigation enhance security and user experience. The system is not only scalable but also adaptable to meet future residential parking needs.

4.2 Use Case Diagram

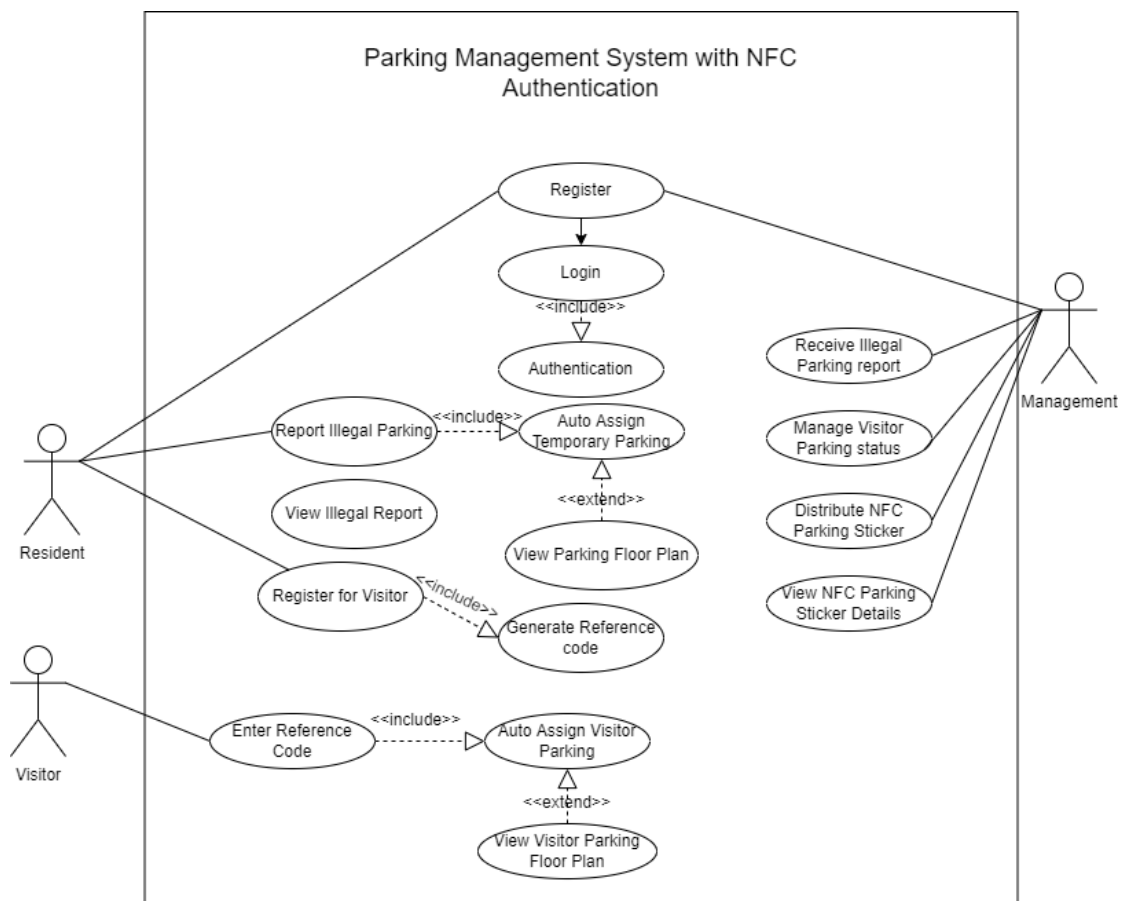


Figure 4.2.1 Use Case Diagram

In this use case diagram, there are three main roles in the parking management system. The three main roles are resident, management, and visitor. The first role is the resident. Before logging into the application, residents must first register an account to proceed. In the resident main activity, there are four key features: reserving visitor parking, reporting illegal parking, viewing NFC parking sticker details, and viewing illegal parking report history.

The first feature is reserving visitor parking. Residents can reserve parking for their visitors by entering the date and time. The system will then generate a reference code, which the resident needs to share with the visitor for verification purposes. The second feature is reporting illegal parking. If a resident finds that someone has parked in their spot or notices illegal parking, they can report it using the app. The management team will receive the illegal parking report immediately. After reporting the illegal parking, the resident has the option to request a temporary parking slot. If requested, the system will automatically assign a temporary slot and display the parking floor plan. Additionally, in the resident main activity, residents can view the illegal parking reports they have submitted. Lastly, residents can view the NFC parking sticker details using their mobile phone's NFC feature. However, residents are only granted access to view the NFC parking sticker details and cannot edit or delete them.

The second role is management. Similar to residents, management must log in to the system and undergo authentication via Firebase before accessing the management main activity. In the management main activity, there are five features: receiving illegal parking reports, managing visitor parking status, writing NFC parking sticker data, updating NFC parking sticker details, and viewing NFC parking stickers.

The first feature is receiving illegal parking reports. Management will receive these reports immediately after a resident submits them, allowing them to arrange security actions, such as clamping the vehicle. The second feature is managing visitor parking status. Management can oversee visitor parking, as some parking spots might

be rented out to residents who need additional parking spaces, requiring management to handle the status of visitor parking.

The third feature is writing NFC parking sticker data. The management is responsible for writing data to these stickers, and the system must be able to write the car details onto the NFC tag. This ensures that during checks or sticker verifications, the NFC tag can display the correct details for the respective car. The fourth feature is updating NFC parking sticker details. This feature is designed to address special cases where the NFC parking sticker needs updating, such as correcting input errors or changing the car's color. Lastly, management can also view NFC parking stickers for verification purposes. This helps to prevent the use of fake stickers or illegal sticker transfers.

The last role is visitor. Visitors do not need to register an account or log in. They will access the visitor main activity through anonymous authentication. Visitors are not required to fill in any personal details. Instead, they are required to enter a reference code provided by a resident to verify their status as a legitimate visitor. Once the code is entered and validated, the system will automatically assign a visitor parking spot and visitor is optional to display the floor plan for navigation purposes.

4.3 Activity Diagram

4.3.1 Login Activity Diagram

The login activity diagram illustrates the step-by-step interaction between the user and the system during the login process in the parking management mobile application. This process ensures that only registered users with valid credentials are granted access to the system.

The sequence begins when the user opens the application. At this point, the system responds by displaying the login interface, which includes fields for email and password input. The user then enters their login information into the form provided. Once the input is submitted, the system takes over by first checking whether the format of the credentials is valid. This includes verifying that the email is in the correct format and the password meets basic requirements.

If the format is valid, the system proceeds to compare the entered credentials with the records stored in the database. A decision is then made based on whether the credentials match any existing account. If a match is found and the credentials are verified as valid, the system displays the main menu and navigates the user to their appropriate homepage based on their role.

However, if the credentials are incorrect either due to a typo, an unregistered email, or a wrong password, the system does not allow access. Instead, it shows an error message, informing the user that the login attempt was unsuccessful. The user is then returned to the login screen, where they can try again.

This login flow ensures both security and usability by providing immediate feedback and guiding the user clearly through the process. It helps protect the system from unauthorized access while maintaining a user-friendly experience for legitimate users.

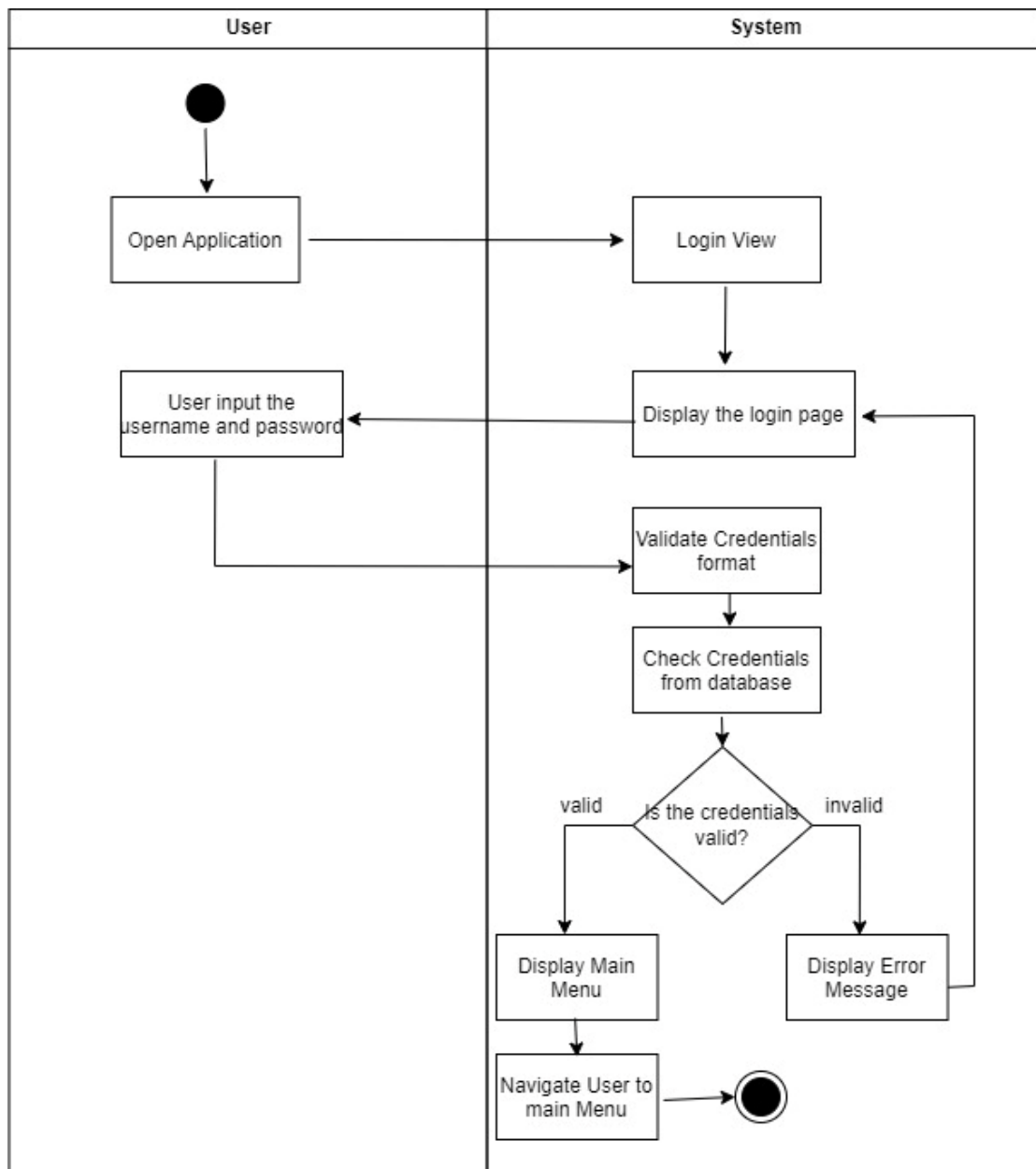


Figure 4.3.1 Login Activity Diagram

4.3.2 Sign Up Activity Diagram

The Sign-Up activity diagram illustrates the step-by-step interaction between the user and the system during the account registration process in the parking management mobile application. This process is designed not only to collect user credentials but also to ensure account authenticity through email verification before granting access to the system.

The process begins when the user opens the application. From the login screen, the user taps on the “Register Here” link to create a new account. In response, the system navigates to and displays the registration page.

Once on the sign-up screen, the user is prompted to input their details, including a username, email address, and password. After submission, the system validates the input by checking for empty fields, proper email format, and password strength.

If the entered details are invalid, such as a missing field or improperly formatted email, the system displays an error message and halts the registration. However, if the details are valid, the system proceeds to send a verification email to the user’s provided email address.

At this stage, the registration is incomplete until the user verifies their email. The system waits for confirmation that the email link has been clicked. Once verified, the account is marked as active, and the user is redirected back to the login page. Only after email verification is complete can the user log in successfully.

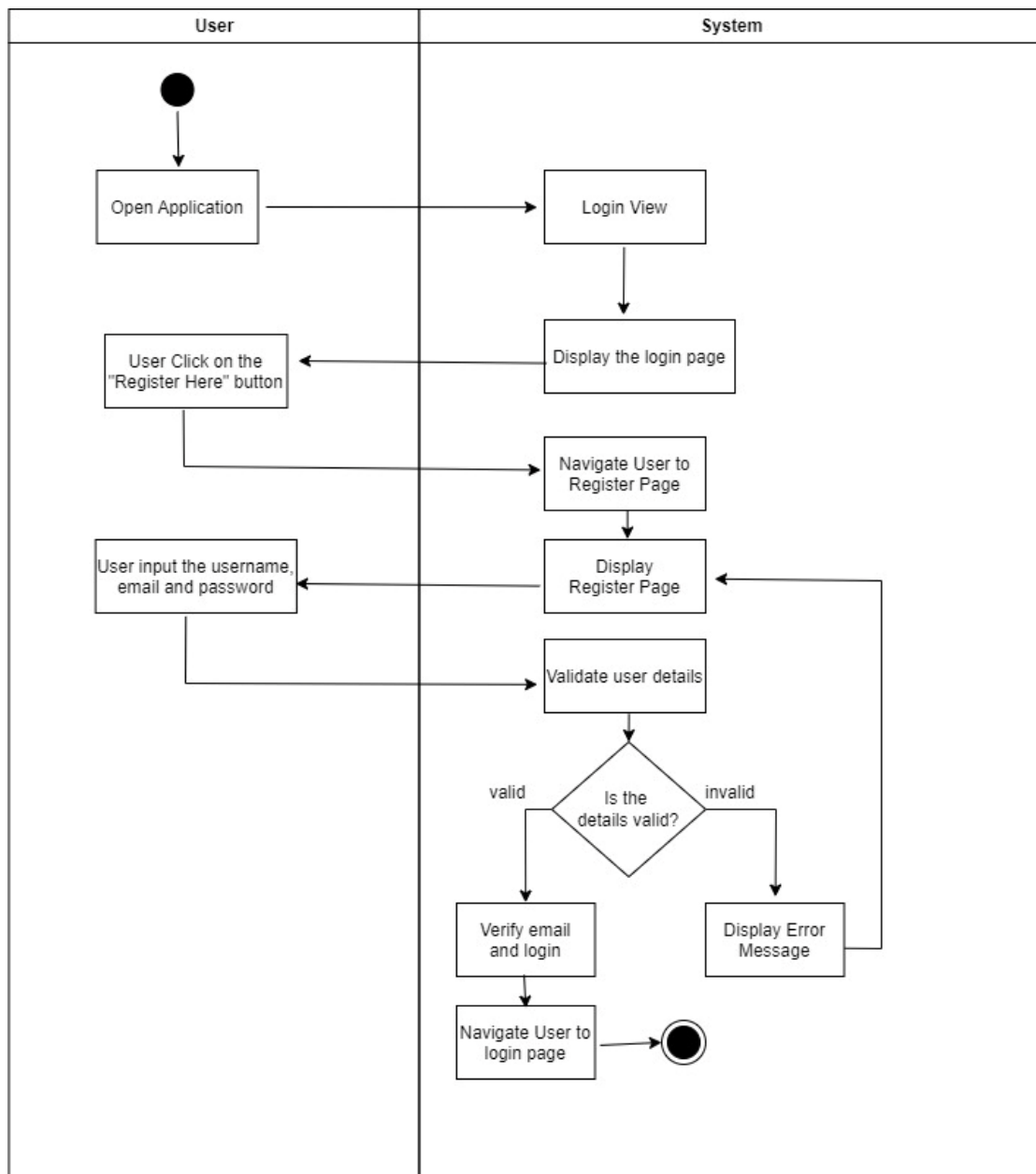


Figure 4.3.2 Sign Up Activity Diagram

4.3.3 Forgot Password Activity Diagram

The Forgot Password activity diagram represents the process users follow when they cannot remember their login credentials and need to reset their password securely. This flow ensures that only valid and registered users are allowed to reset their passwords using their verified email addresses.

The process starts when the user opens the application and is directed to the login page. If the user forgets their password, they can tap on the “Forgot Password?” link. This action prompts the system to navigate the user to the Forgot Password screen.

On this screen, the user is asked to input the email address associated with their account. Once the email is submitted, the system validates the format and checks if the email exists in the database. If the email is invalid or not registered, the system displays an error message and prompts the user to try again.

However, if the email is valid and registered in the system, the application proceeds to send a reset password link to the provided email address. The user then accesses their email and clicks on the link, which allows them to enter a new password. Once the new password is submitted, the system updates the user’s information in the database.

After the update, the user is allowed to log in using their new password. This process ensures both security and convenience, making it easy for users to recover their accounts while protecting against unauthorized access.

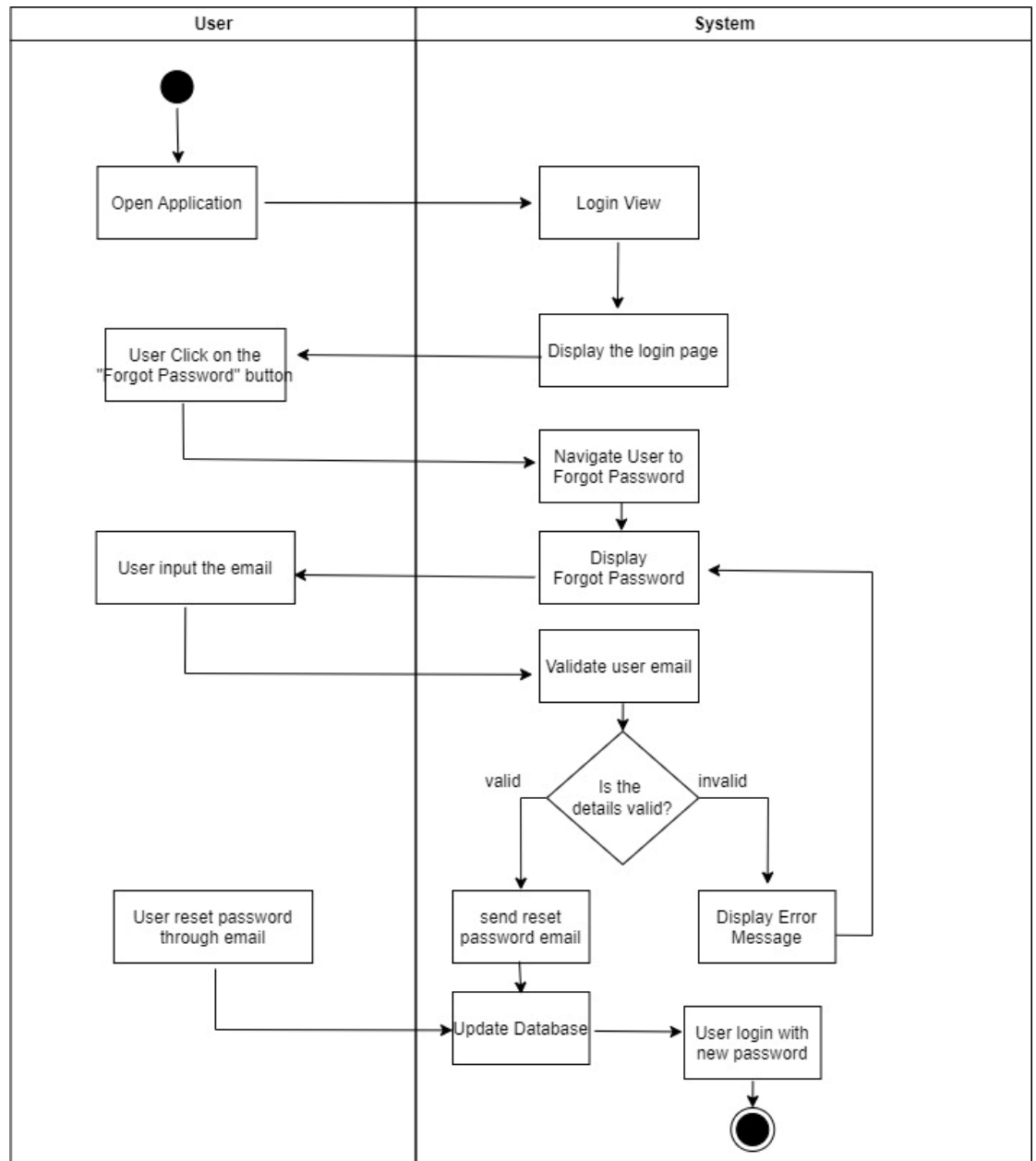


Figure 4.3.3 Forgot Password Activity Diagram

4.3.4 Visitor Main Page

The activity diagram for the Visitor Main Page outlines the process that enables visitors to access a temporary parking space using a reference code provided by residents. This flow ensures that only invited visitors are granted access and automatically assigned a parking lot within the system.

The process begins when the user opens the application and is presented with the login screen. Instead of entering login credentials, the visitor clicks on the “I’m

Visitor” button. Upon this action, the system navigates the user to the Visitor Main Page where they are prompted to input a reference code.

The reference code is a unique identifier provided by the resident who registered the visitor. After entering the code, the system validates it by checking it against the stored records in the Firebase database. If the reference code is found to be invalid or already used, the system displays an error message, prompting the visitor to check and re-enter the correct code.

On the other hand, if the reference code is valid and unused, the system proceeds to the auto-assignment module. This module automatically checks for available parking spots and assigns one to the visitor. Once assigned, the system displays the designated parking slot on the screen, completing the visitor’s parking registration process.

This process is designed to be secure, efficient, and user-friendly, ensuring that only authorized visitors can access parking and helping to maintain proper access control within the residential premises.

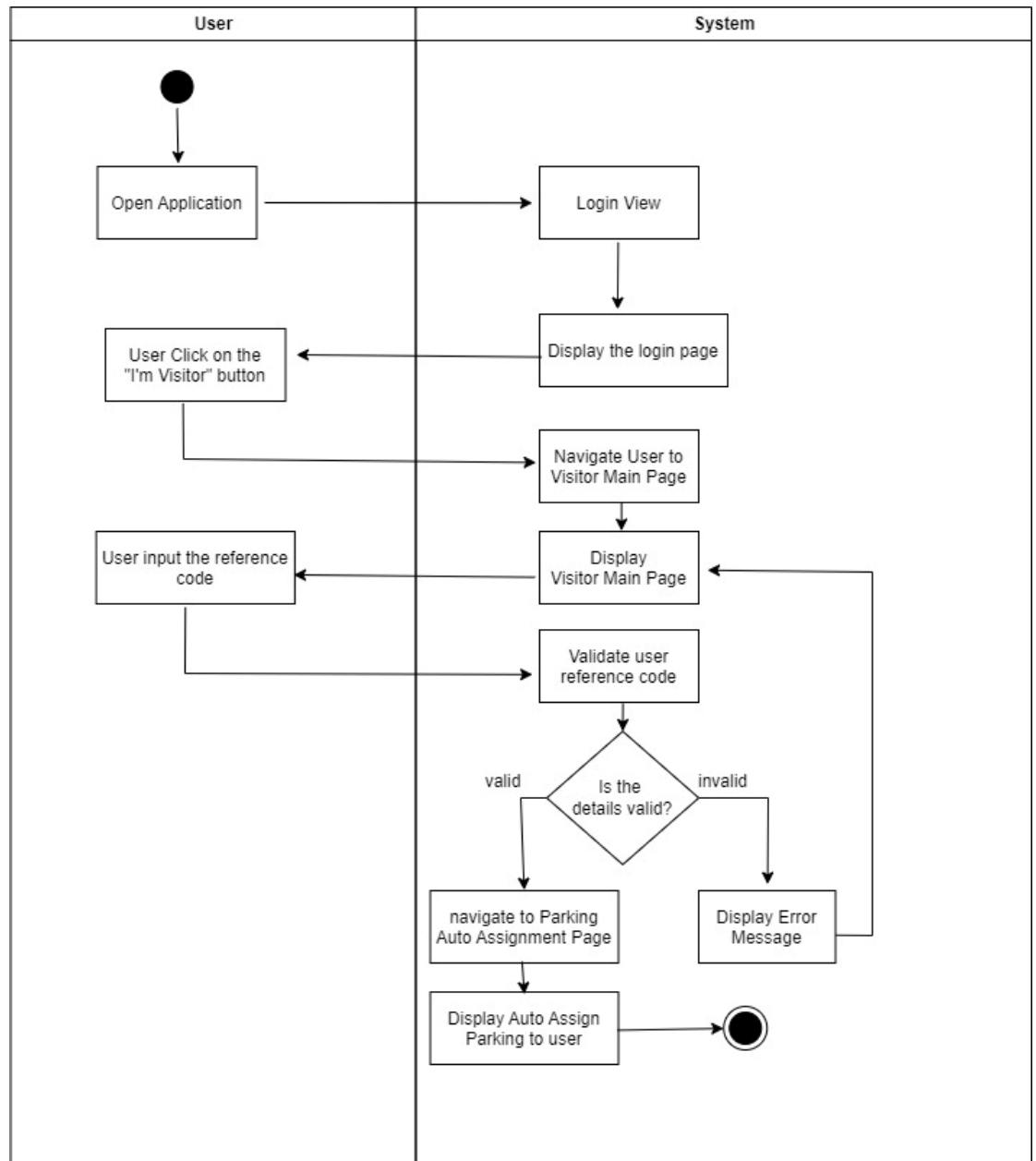


Figure 4.3.4 Visitor Main Activity Diagram

4.3.5 Auto Assignment Parking Activity Diagram

The Auto Assign Parking activity diagram outlines the process through which visitors are granted a parking spot using a reference code provided by a resident. This workflow ensures that the system securely validates the visitor's authorization and helps them navigate to the assigned parking location.

The process begins after the user navigates to the Visitor Main Page. From there, the user inputs the reference code, which was generated and shared by the resident during the visitor registration process. Once entered, the system performs a validation check to ensure the code is both correct and has not been previously used.

If the reference code is invalid, the system immediately responds by displaying an error message, prompting the user to re-enter the correct code. However, if the reference code is valid, the system proceeds to the next step—automatically assigning the visitor a parking spot. This is done by checking the availability of parking spaces in real-time, starting from Level 1 and moving to other levels if necessary.

Once a spot is successfully assigned, the system displays the assigned lot to the visitor. The user then has the option to view a visual parking map by clicking on the “View Map” button. This map highlights the assigned lot and shows directional arrows from the entrance to help the user navigate easily, even in complex or multi-level parking structures.

Finally, after reviewing the parking information, the user can click the “Exit” button to return to the main interface or close the session. This activity ensures a smooth, efficient, and secure experience for visitors seeking parking in the facility.

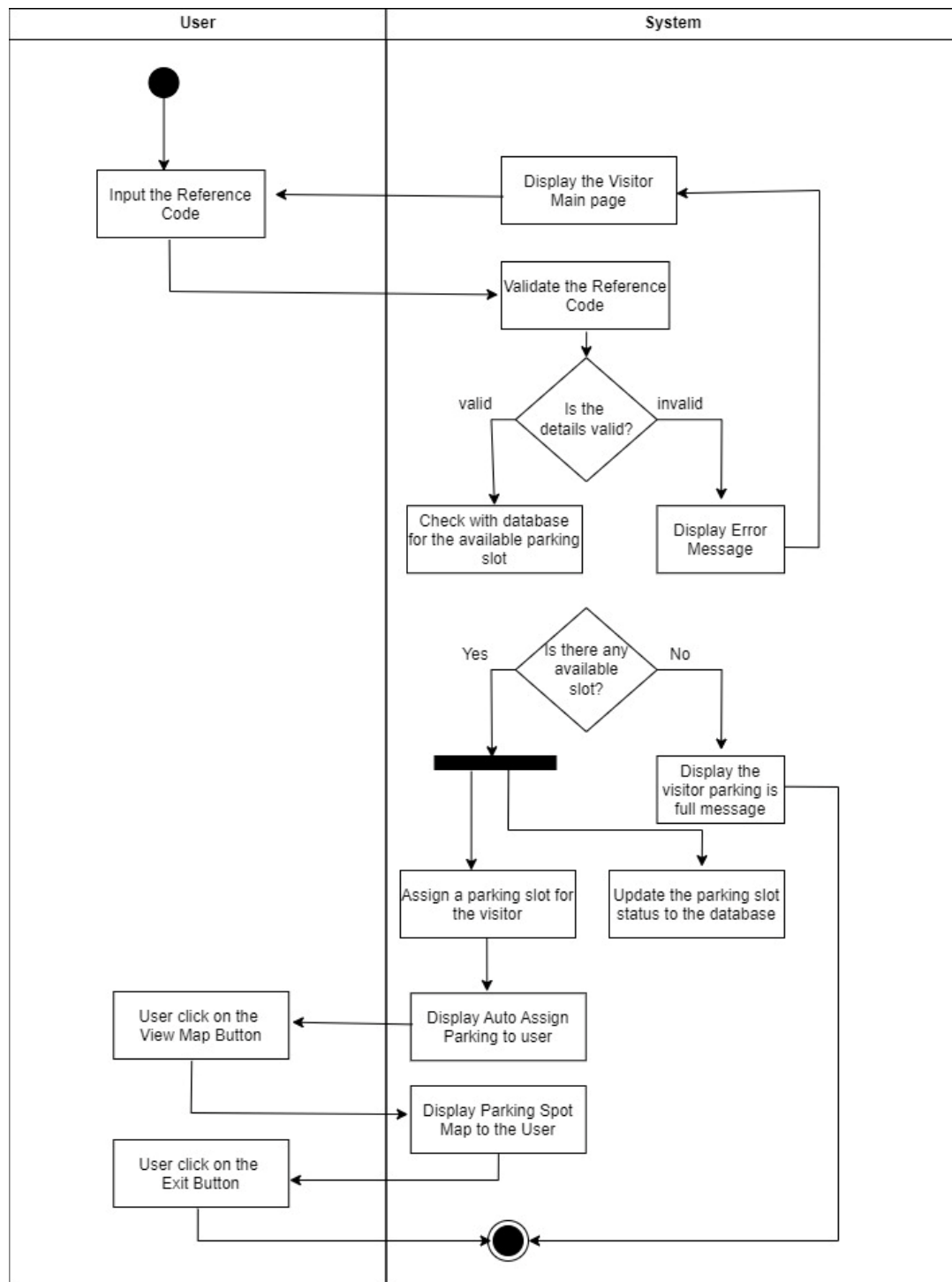


Figure 4.3.5 Auto Assignment Parking Activity Diagram

4.3.6 Report Illegal Parking

The activity diagram for the “Report Illegal Parking” module represents the process a resident follows to submit a complaint about an unauthorized vehicle

occupying a parking space. This feature plays a crucial role in maintaining parking discipline within the community and enables swift action by the management.

The process begins when the resident logs into their account and lands on the Resident Main Menu. From there, the user clicks the “Report Illegal Parking” button, which directs them to the reporting interface. The system navigates to the appropriate page and displays the report form for the resident to fill in.

Once the form is presented, the resident inputs the necessary report details. These typically include the vehicle’s license plate, the location, a description of the violation, and an image as photographic evidence. After filling in the required fields, the information is submitted to the system for validation.

At this point, the system checks whether all required fields have been correctly filled. If any details are missing or invalid, the system displays an error message and prompts the user to correct the inputs.

If the details are valid, the system proceeds to submit the report to Firebase Firestore for backend storage. Upon successful submission, the system confirms the report with a success message, assuring the resident that the complaint has been received and will be processed by the management.

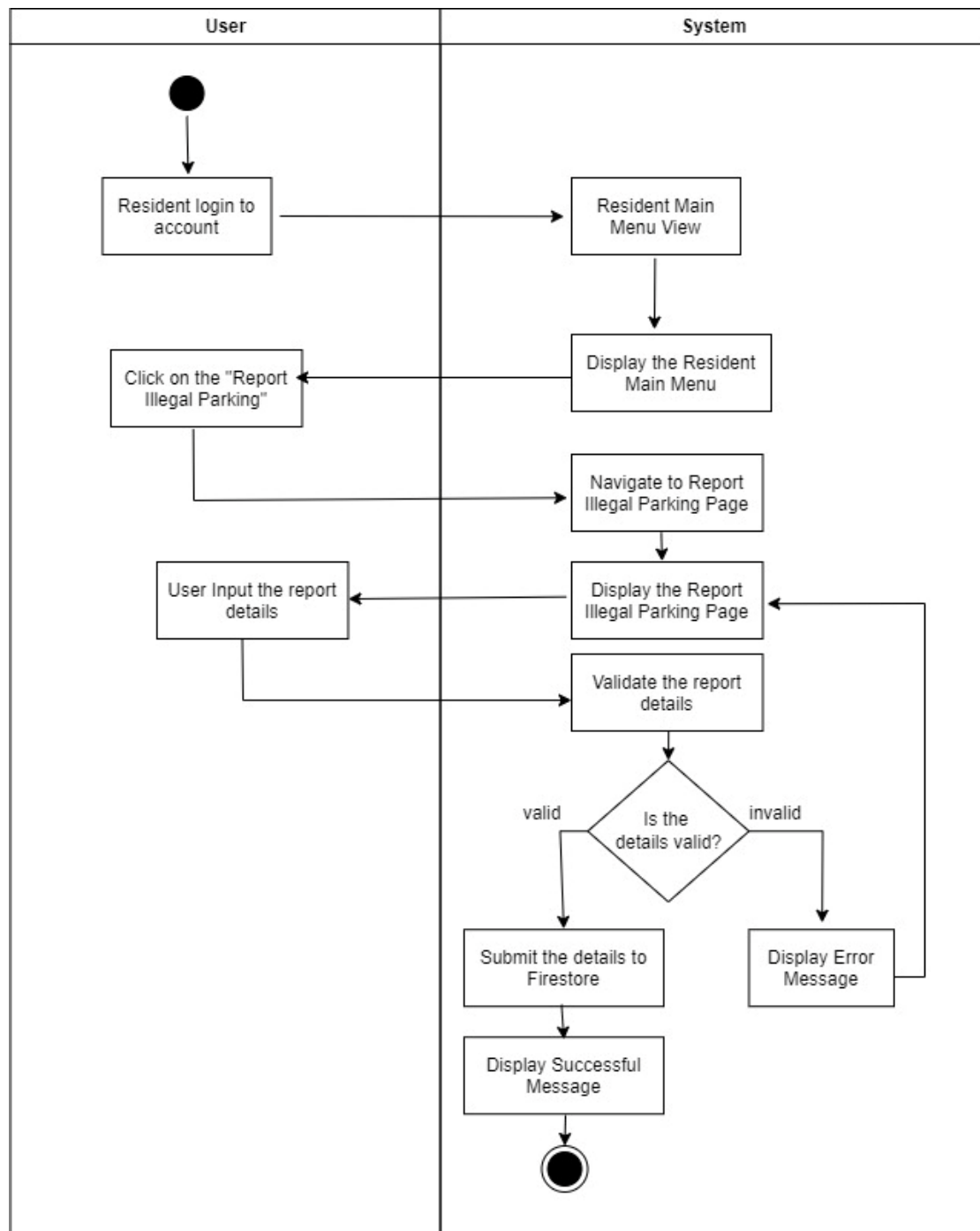


Figure 4.3.6 Report Illegal Parking Activity Diagram

4.3.7 View My Reports

The “View My Reports” activity diagram outlines the steps a resident follows to access and review the status of their submitted illegal parking complaints. This feature is essential in promoting transparency and ensuring that residents stay informed about how their concerns are being handled by the management.

The process begins once the resident successfully logs into their account and arrives at the Resident Main Menu. From there, the user selects the “View My Reports” option. This action prompts the system to navigate to the dedicated report viewing page, where all reports submitted by that resident are retrieved and displayed.

Upon entering the View My Reports page, the system loads the report list from Firestore. Each report includes details such as the car plate number, location, timestamp, report description, photo evidence, and most importantly, the current status of the case—whether it is Pending, In Progress, or Resolved.

This module is designed to ensure that residents only have access to their own reports. It respects user privacy by filtering data based on the currently logged-in account, displaying only the records tied to the resident’s unique identifier (such as email).

Once the resident has finished viewing the report details, they can press the Back button. The system will then navigate the user back to the Resident Main Page, maintaining a smooth and user-friendly experience.

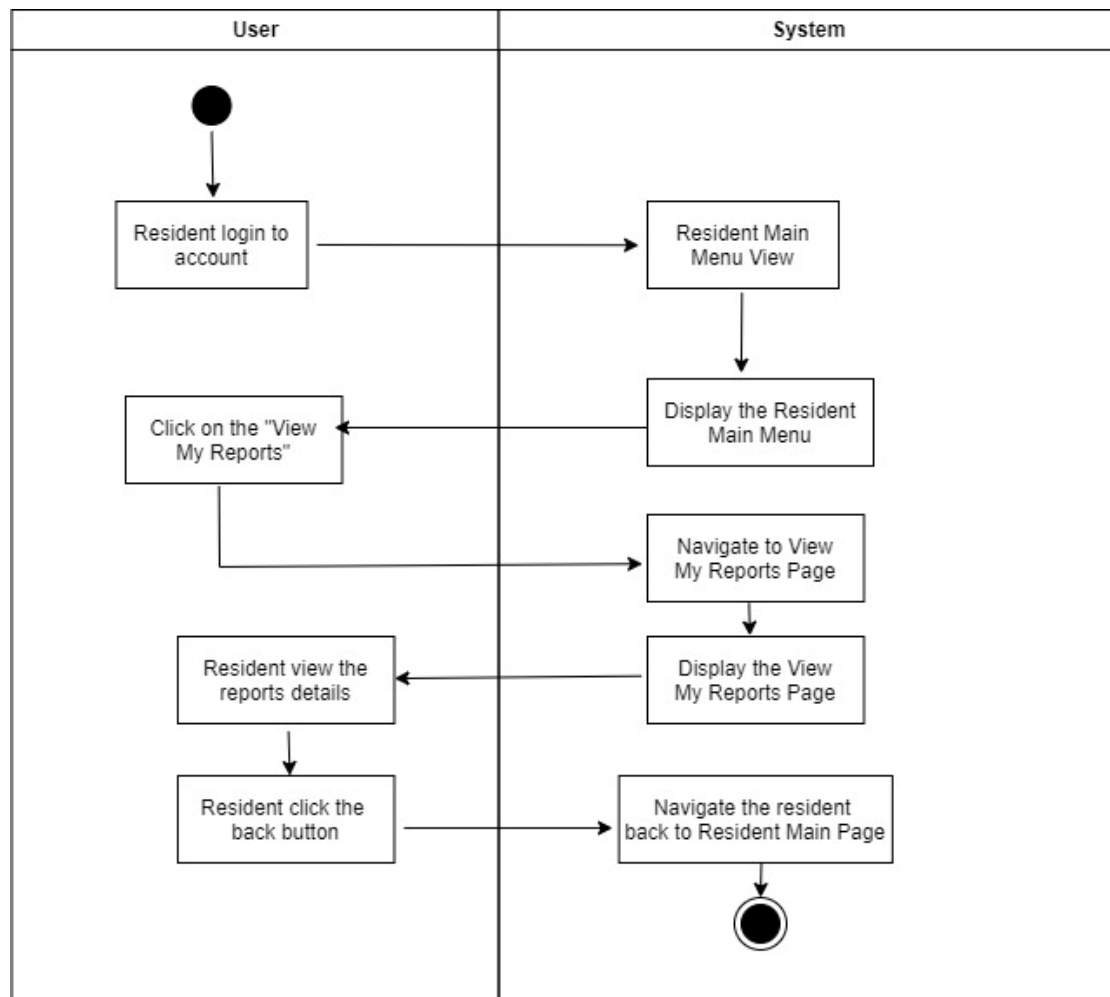


Figure 4.3.7 View My Reports Activity Diagram

4.3.8 Register Visitor Activity Diagram

The "Register Visitor" activity diagram illustrates the step-by-step interaction between a resident and the system when registering a visitor in the parking management application. This feature is designed to ensure that only authorized guests are allowed entry, enhancing both security and visitor tracking.

The process begins once a resident logs in to their account and reaches the main menu. From there, the resident selects the "Register Visitor" option, prompting the system to navigate to the registration page. The system then displays the registration form where the resident is required to fill in the visitor's full name, identity card number, car plate number, and phone number.

Once the form is submitted, the system performs a validation check to ensure all inputs meet the required format and that no fields are left empty. If the data is invalid, an error message is displayed, prompting the resident to correct the input. If the data passes validation, the system proceeds to store the visitor's details in Firestore.

After a successful submission, the system automatically generates a unique reference code tied to the registered visitor. This code acts as a secure verification token that the visitor must later provide to gain access and receive a parking space.

The resident is presented with two action buttons: "Copy" and "Done." The "Copy" button allows the resident to easily copy the reference code to their clipboard for sharing with the visitor, while the "Done" button completes the process and navigates the resident back to the main menu.

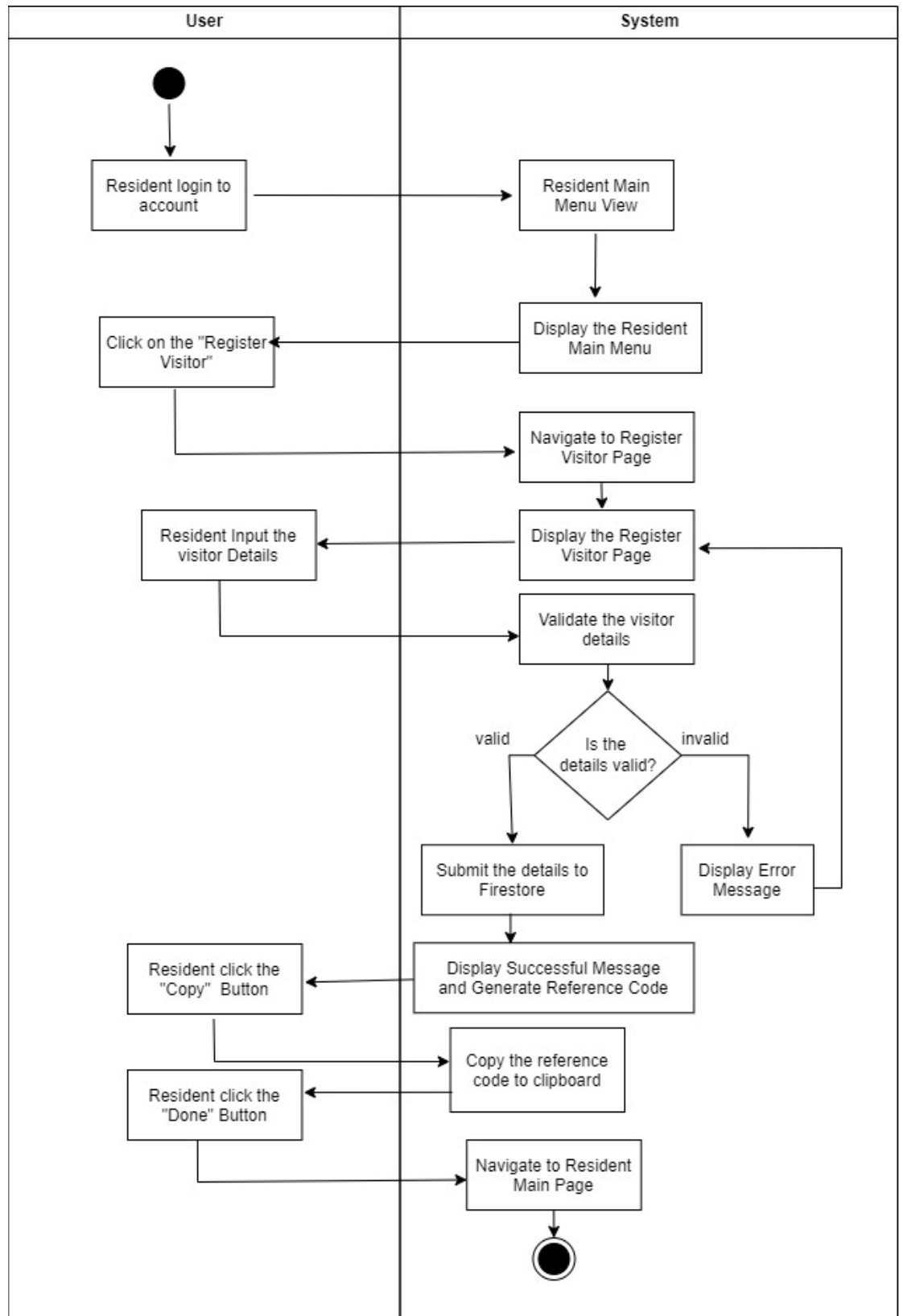


Figure 4.3.8 Register Visitor Activity Diagram

4.3.9 Manage Illegal Parking

The activity diagram for managing illegal parking reports outlines how the management team interacts with the system to handle and respond to resident-submitted complaints. This process is crucial in maintaining order within the parking facility and ensuring that all reports are addressed promptly and transparently.

The flow begins when a management user logs in and is redirected to the Management Main Menu. From there, they select the “Illegal Parking Reports” option, which prompts the system to navigate to the relevant page and retrieve all available reports from Firestore, the cloud database used by the application.

Once the reports are loaded, the management user can view the report details, which typically include the car plate number, location, timestamp, violation description, and any accompanying photo evidence. After reviewing the information, the user is given the ability to update the status of each report—choosing between options such as "Pending", "In Progress", or "Resolved" depending on the progress of the case.

Upon selecting a new status, the system updates the record in Firestore and immediately retrieves the latest version to reflect the changes. This real-time synchronization ensures that both management and residents are always presented with the most up-to-date information.

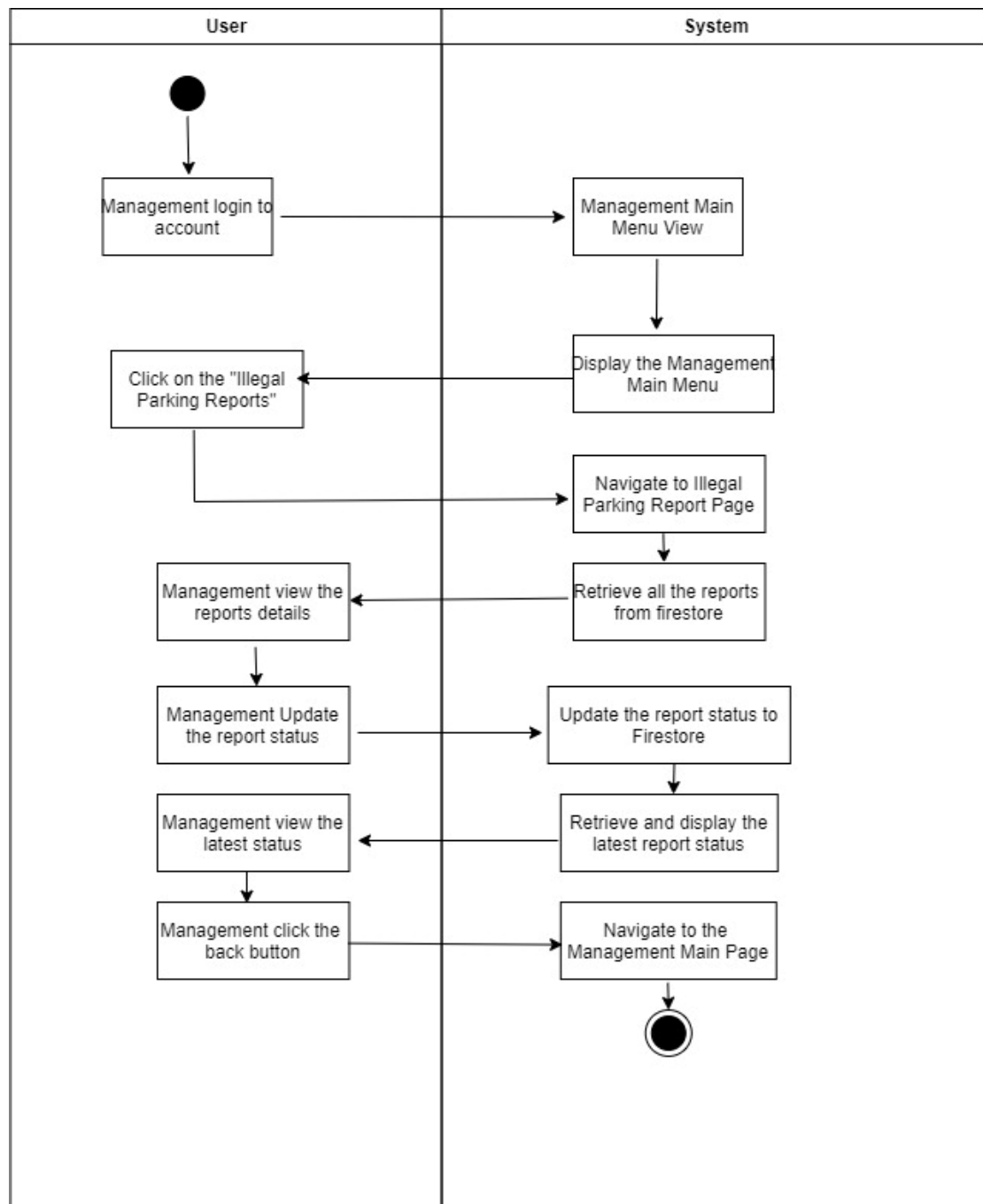


Figure 4.3.9 Manage Illegal Parking Activity Diagram

4.3.10 NFC Tag Writer

The NFC Tag Writer activity diagram illustrates the process by which management securely writes resident parking information to an NFC tag. This feature plays a key role in enhancing parking lot security by ensuring only authorized vehicles gain access based on tag verification.

The process begins when a management user logs into their account and accesses the Management Main Menu. From there, they click on the “Write NFC Tag” option, which navigates the system to the NFC Car Info Page. On this page, management is prompted to input car details, including the license plate number, vehicle model, colour, and assigned parking lot.

Once the car details are entered, the system validates the information. If the input is incomplete or invalid, an error message is displayed and the process stops. Otherwise, the system proceeds to the NFC Tag Writer Page and also stores the car details in Firebase (Firestore) for backup and verification purposes.

At this point, the management staff is instructed to tap an NFC tag to the mobile device. When the tag is detected, the system attempts to write the car data to it. If the write process fails, an error message is displayed and the user can retry. If successful, the application navigates to a Success Page that displays the written details and confirms the write was successful.

Finally, after reviewing the success confirmation, the user clicks a back button to return to the main dashboard. This flow ensures the NFC tag is programmed only after successful data validation and provides clear feedback for every outcome, helping management maintain secure and accurate control of parking permissions

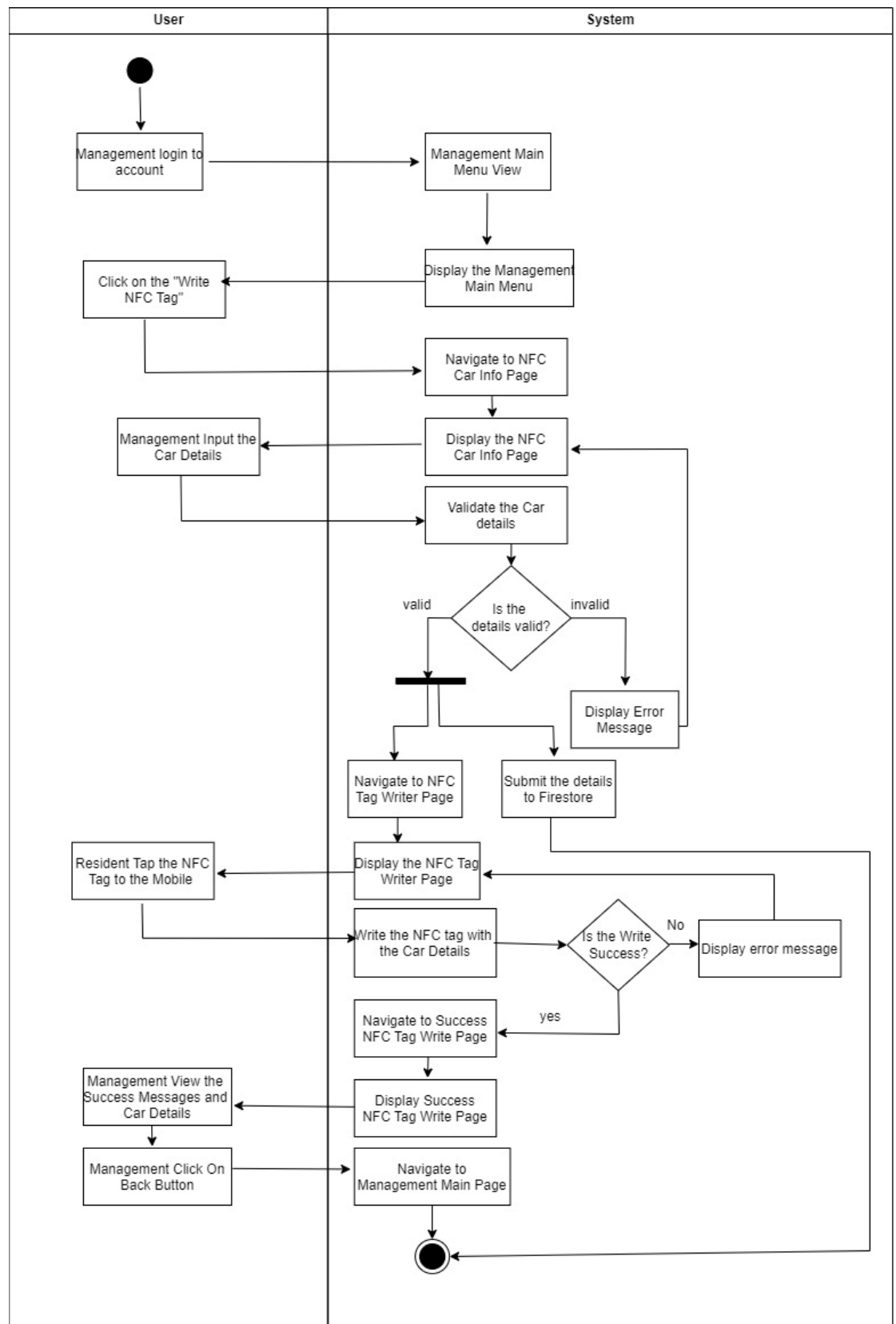


Figure 4.3.10 NFC Tag Writer Activity Diagram

4.3.10 NFC Tag Reader

The NFC Tag Reader activity diagram illustrates the process by which management staff can verify parking details during routine patrols. This feature is designed to improve the efficiency of inspections and ensure that vehicles are parked in their assigned spaces based on authenticated NFC tags.

The process begins when a management user logs into the system and accesses the Management Main Menu. From this screen, they select the "Read NFC Tag" option, which navigates them to the NFC Tag Reader Page. At this point, the system is ready to scan an NFC tag.

Management then taps a vehicle's NFC tag using a mobile device equipped with NFC capability. Once tapped, the system attempts to read the NFC tag data. The read process checks if the tag contains valid and recognizable information. If the tag is unreadable or invalid, the system displays an error message, prompting the user to try again or investigate further.

If the NFC tag is valid, the application navigates to the NFC Tag Reader Success Page, where all the stored vehicle details—such as license plate, color, model, and assigned parking lot—are displayed. This allows management to quickly verify that the car is parked correctly and the tag has not been tampered with.

After reviewing the information, the management user clicks the back button to return to the main menu, ready to continue inspections or perform other tasks.

This activity supports fast, secure, and contactless verification during patrols, enhancing both convenience and enforcement in parking management.

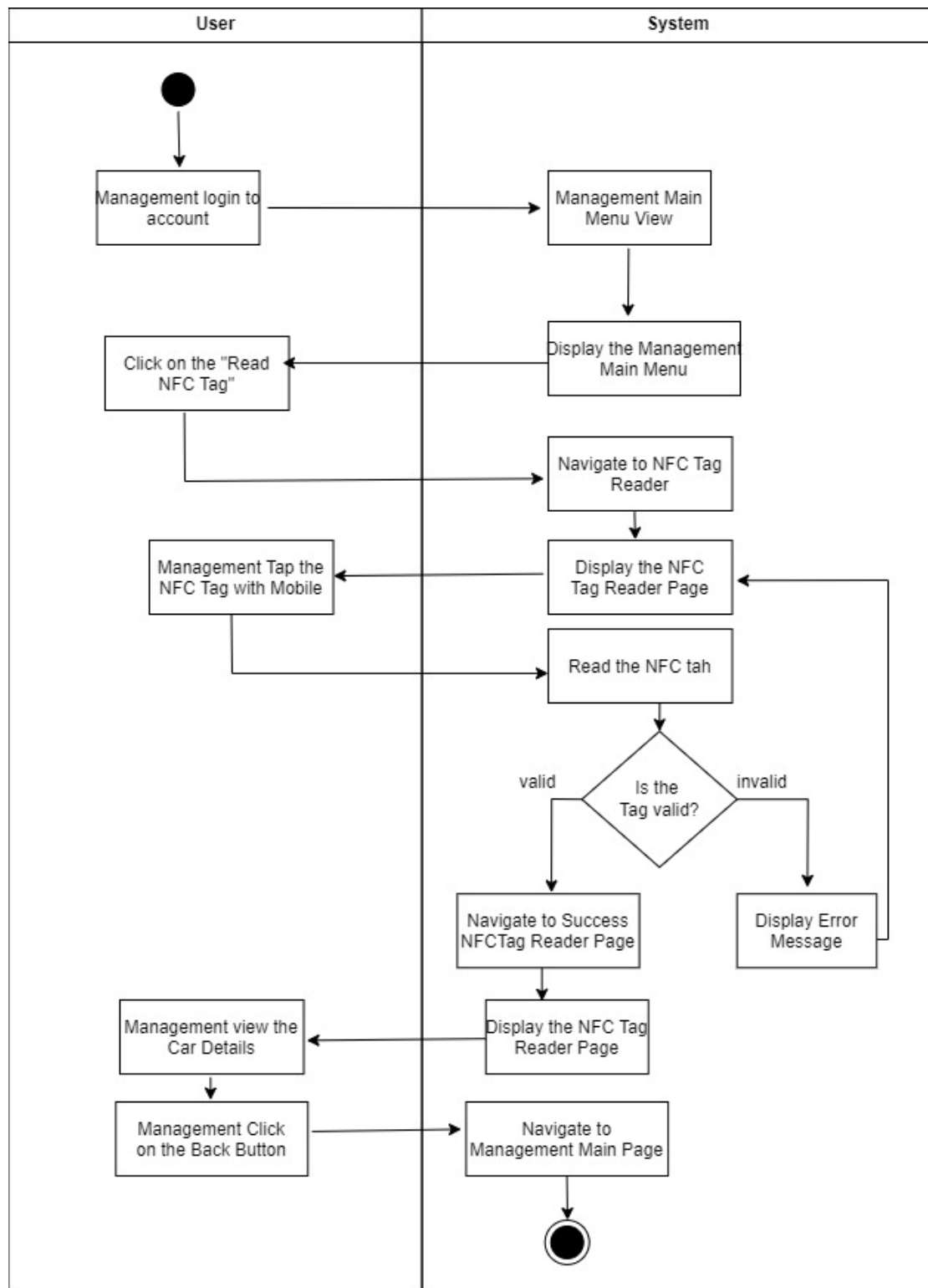


Figure 4.3.11 NFC Tag Reader Activity Diagram

4.4 Database Design

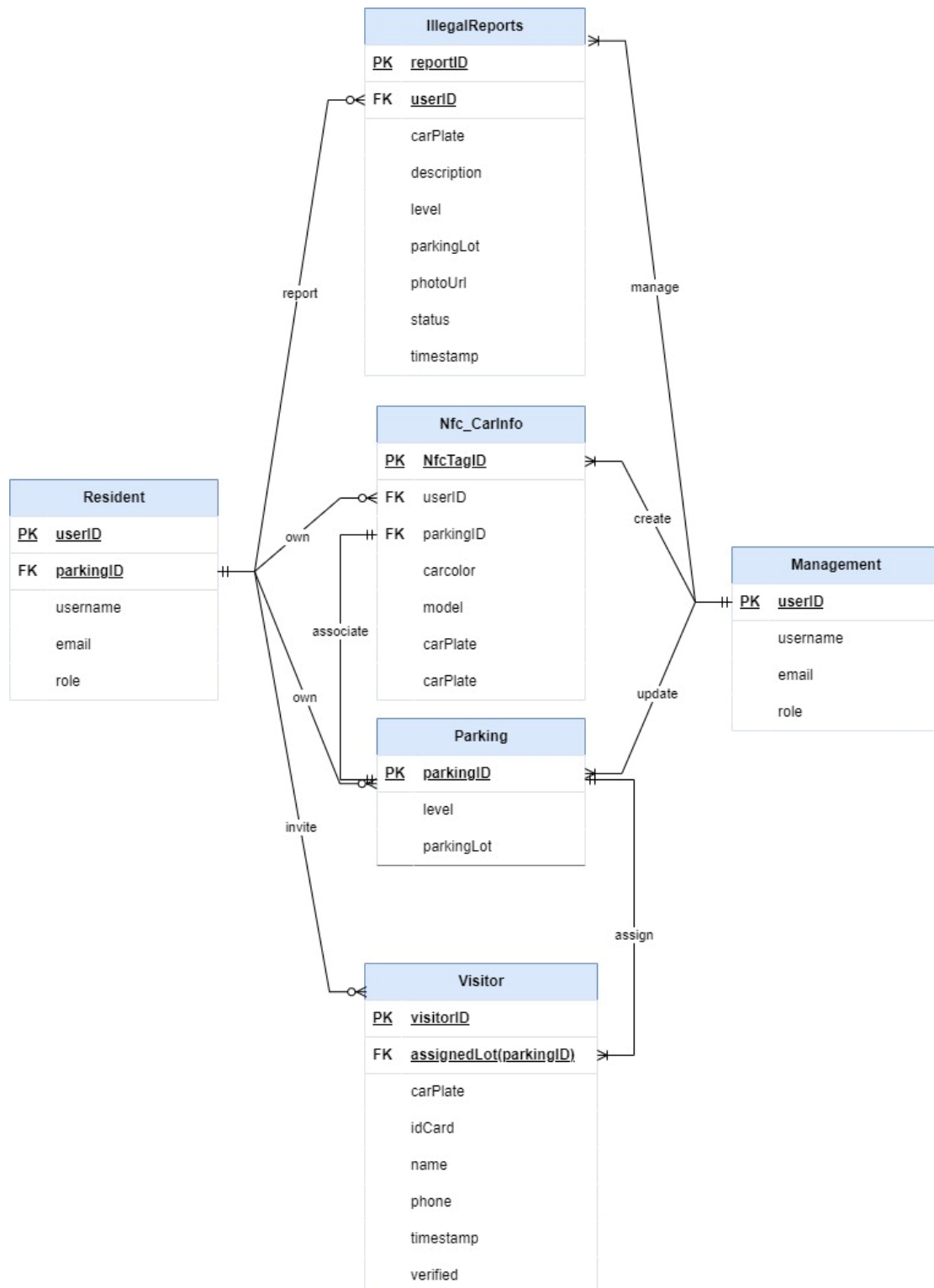


Figure 4.4.1 Database Design

The database design of the Parking Management System with Mobile NFC Authentication plays a crucial role in ensuring data integrity, user-role segregation, visitor validation, and proper parking space allocation using NFC authentication. The Entity-Relationship Diagram (ERD) illustrates the interaction between six main entities: Resident, Visitor, Management, Parking, Nfc_CarInfo, and IllegalReports.

1. Resident

- A Resident owns a specific parking lot (parkingID) and can invite Visitor records.
- A resident is also associated with an NFC tag that represents their vehicle.

| Field Name | Data Type | Null | FK/PK | Description |
|------------|-----------|------|-------|---|
| userID | Varchar | No | PK | Unique identifier for the resident. |
| parkingID | Varchar | No | FK | Foreign key referencing assigned parking lot. |
| username | Varchar | No | | Resident's full name or display name. |
| email | Varchar | No | | Resident's registered email address. |
| role | Varchar | No | | User role (e.g., 'resident'). |

Table 4.4.1 Resident Table

2. Management

- Management users are responsible for creating or updating NFC tag and parking data.
- They also manage and review illegal parking reports.

| Field Name | Data Type | Null | FK/PK | Description |
|------------|-----------|------|-------|------------------------------------|
| userID | Varchar | No | PK | Unique identifier for the manager. |

| | | | | |
|----------|---------|----|--|---------------------------------|
| username | Varchar | No | | Manager's name. |
| email | Varchar | No | | Manager's email address. |
| role | Varchar | No | | User role (e.g., 'management'). |

Table 4.4.2 Management Table

3. Visitor

- Visitors are registered by residents and assigned a specific parking lot.
- The system verifies the visitor using a reference code, typically corresponding to the visitorID, and assigns the parking lot upon verification.

| Field Name | Data Type | Null | FK/PK | Description |
|------------|-----------|------|-------|--|
| visitorID | Varchar | No | PK | Unique identifier for each visitor. |
| parkingID | Varchar | No | FK | Foreign key referencing the assigned parking lot. |
| carPlate | Varchar | No | | Visitor's vehicle license plate number. |
| idCard | Varchar | No | | Visitor's identification card number. |
| name | Varchar | No | | Visitor's full name. |
| phone | Varchar | No | | Contact number of the visitor. |
| timestamp | TimeStamp | No | | Time when the visitor was registered. |
| verified | boolean | No | | Whether the visitor's reference code has been verified |

Table 4.4.3 Visitor Table

4. Parking

- Represents a physical parking space within a specified level.
- Each parking space is uniquely linked to an NFC tag and may be assigned to either a Resident or a Visitor.

| Field Name | Data Type | Null | FK/PK | Description |
|------------|-----------|------|-------|---|
| parkingID | Varchar | No | PK | Unique identifier for the parking lot. |
| level | Varchar | No | | Level/floor where the parking lot is located. |
| parkingLot | Varchar | No | | Identifier/label for the specific lot (e.g., A102). |

Table 4.4.4 Parking Table

5. IllegalReports

- This entity records reports submitted by residents regarding unauthorized or illegal parking incidents.
- Includes detailed information and optionally a photo as evidence.
- Managed by the Management role, which oversees and changes the report status.

| Field Name | Data Type | Null | FK/PK | Description |
|-------------|-----------|------|-------|--|
| reportID | Varchar | No | PK | Unique identifier for the report |
| userID | Varchar | No | FK | Foreign key referencing the resident who reported. |
| carPlate | Varchar | No | | Plate number of the vehicle violating parking rules. |
| description | Text | No | | Detailed description of the violation. |
| level | Varchar | No | | Level/floor where the violation occurred. |
| parkingLot | Varchar | No | | Specific parking lot involved in the report. |
| photoUrl | Text | No | | URL link to the image evidence (stored in cloud). |

| | | | | |
|-----------|-----------|----|--|---|
| status | Varchar | No | | Report status (e.g., 'Pending', 'In Progress', 'Resolved'). |
| timestamp | TimeStamp | No | | Time when the report was submitted. |

Table 4.4.5 Illegal Report Table

6. Nfc_CarInfo

- This entity stores data associated with an NFC tag.
- Each NFC tag is uniquely assigned to one user (usually a resident) and one parking lot.
- It includes vehicle-specific information such as color, model, and plate number.
- The relationship ensures that one parking lot is associated with one NFC tag to maintain accurate allocation.

| Field Name | Data Type | Null | FK/PK | Description |
|------------|-----------|------|-------|---|
| NfcTagID | Varchar | No | PK | Unique identifier for the NFC tag. |
| userID | Varchar | No | FK | Foreign key referencing the car owner (resident). |
| parkingID | Varchar | No | FK | Foreign key referencing the parking lot of the car. |
| carcolor | Varchar | No | | Colour of the registered vehicle. |
| model | Varchar | No | | Car model (e.g., Honda Civic). |
| carPlate | Varchar | No | | Vehicle's license plate number. |

Table 4.4.6 Nfc_Carinfo Table

4.5 Concluding Remark

The system design presented in this chapter outlines a comprehensive and secure architecture for the Parking Management System with Mobile NFC Authentication. Through the integration of Android Studio, Firebase, and NFC technology, the design addresses the core requirements of a modern residential parking environment, ensuring accessibility, security, and efficiency for all user roles: management, residents, and visitors.

The system architecture is purposefully structured to provide role-based access and tailored functionalities. Residents can report illegal parking, register visitors, and view their own report history, all through a streamlined interface. Meanwhile, visitors are granted secure access through reference codes and are automatically assigned available parking slots, reducing management overhead and preventing unauthorized entry. The management team, empowered with a suite of features including NFC tag writing and report tracking, can oversee and validate parking enforcement with ease.

The use of Firebase as the backend ensures real-time synchronization, secure data storage, and efficient user authentication, while NFC technology eliminates the possibility of counterfeit parking stickers, significantly improving vehicle verification accuracy. The incorporation of activity diagrams and a detailed entity-relationship model further validates the functional completeness of the design.

Ultimately, this chapter not only documents the structural blueprint of the application but also emphasizes the importance of usability, security, and automation in solving real-world parking challenges. The system is scalable, modular, and adaptable for future enhancements, setting a strong foundation for implementation and deployment in real residential communities.

CHAPTER 5

System Implementation

5.1 Software Setup and Configuration

Android Studio

The development of the parking management application was carried out using Android Studio, a widely used integrated development environment (IDE) for Android application development. Java was selected as the primary programming language for coding the application's logic, while XML was utilized for designing the layout and user interface. This combination of technologies allowed for a structured, efficient, and scalable application architecture. Below are the details of the setup and configuration

Step 1: Download Android Studio

Visit the Android Studio website (<https://developer.android.com/>) and download the application. The version I have downloaded is 2023.1.1.28.

Step 2: Install Android Studio

Execute the installation package, which guided the installation of the IDE along with necessary SDK components, including the Android SDK, SDK tools, and the emulator for testing.

Step 3: Configure SDK and Virtual Devices

The SDK manager in Android Studio was used to install the required Android versions and system images for testing the application across different Android devices.

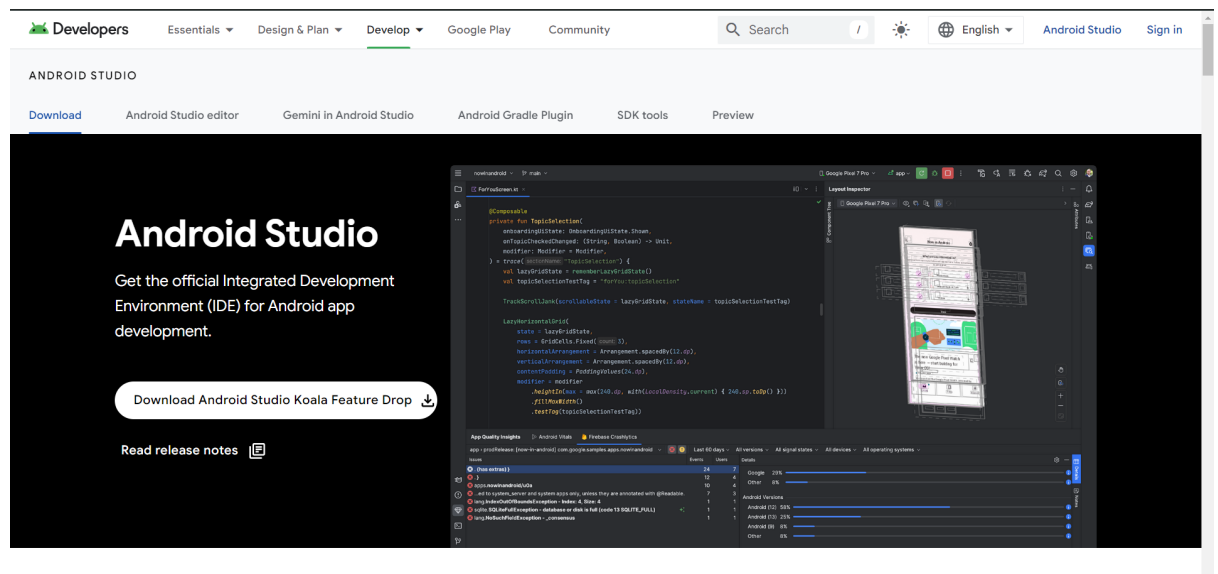


Figure 5.1.1 Android Studio Official Website

Firestore Integration

Firestore was used as the backend solution, it can provide real-time data synchronization, user authentication, and cloud storage. Firestore was chosen for integration with Android because of its suite of cloud services, efficient data management, and scalability for the application.

Step 1: Account and project creation

Visit the firebase website (<https://firebase.google.com/>) and sign in with the google account. Go to console create a project and fill in the details.

Step 2: Integration of Android Studio with Firestore with the

Go to the Android studio create a project or open the existing project. Go for main menu and select the tools, there is a firebase belongs to tools. Select authentication and select the authentication using a custom system. By clicking the connect firebase, it will redirect to the web and select the project created in the firebase. The android studio is connected to firebase. Add the SDK dependencies to the applications. Then, repeat the same process with storage and firestore.

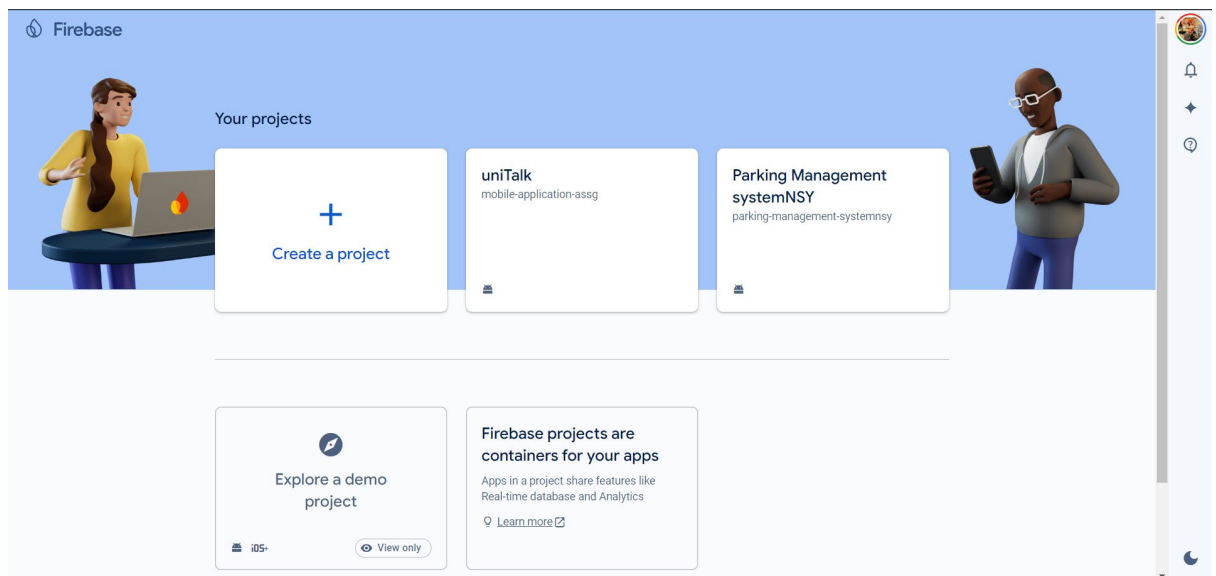


Figure 5.1.2 Firebase Website

5.2 Implementation Details

5.2.1 Implementation Details for Firebase Integration

To facilitate robust backend support for the parking management application, Firebase has been integrated to handle authentication, data storage, and file management. This section delves into the technical aspects of integrating Firebase services, including Firebase Authentication for user management, Firestore for real-time data synchronization, and Firebase Storage for handling file uploads and downloads. By incorporating these components, the application achieves secure user authentication, efficient data handling, and scalable storage solutions. The following subsections detail the specific steps and code implementations necessary to integrate these Firebase services seamlessly into the Android development environment.

1. Firebase Authentication

Step 1: Include the firebase authentication SDK dependencies in the build.gradle (Module:app)

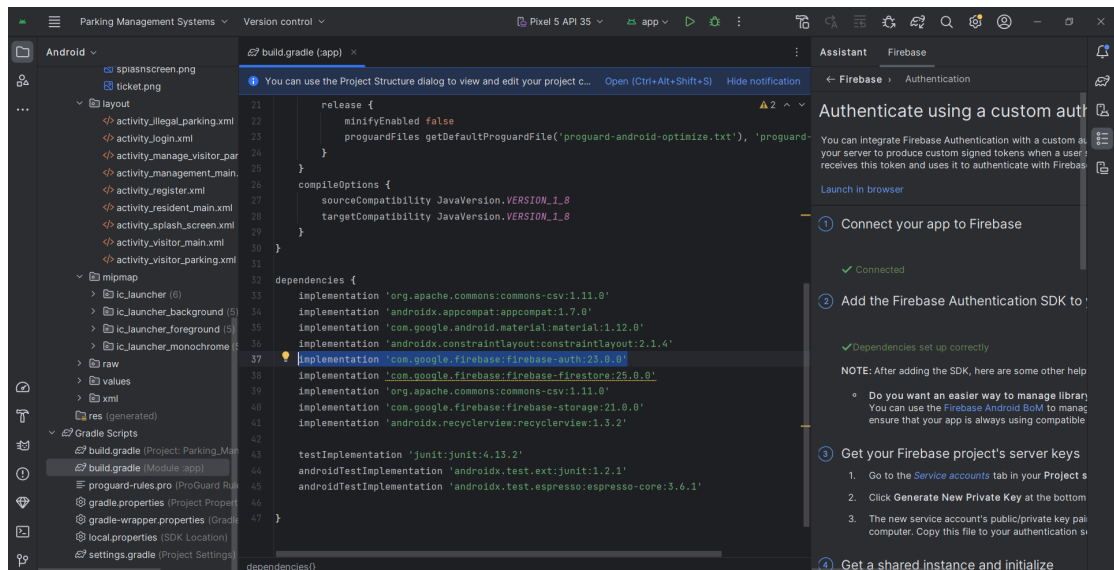


Figure 5.2.1.1 Add the Firebase Authentication SDK Dependencies to build.gradle

Step 2: Sync project with Gradle files.

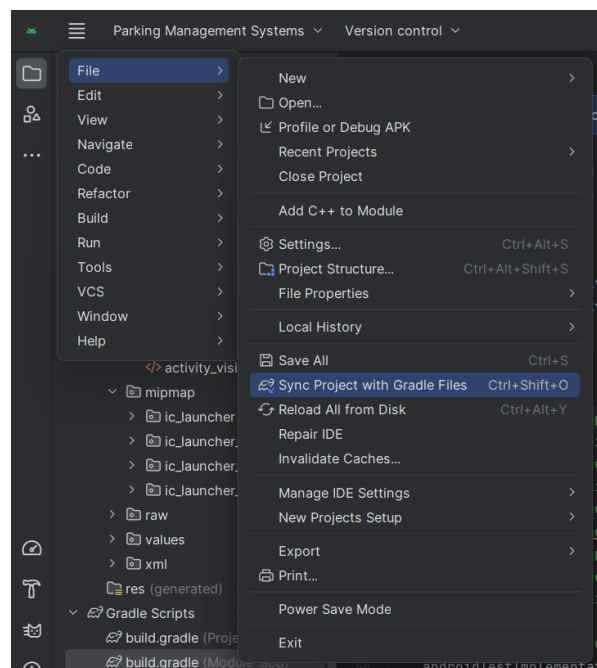


Figure 5.2.1.2 Sync Project with Gradle files.

Step 3: In the Firebase Console, enable the authentication methods.

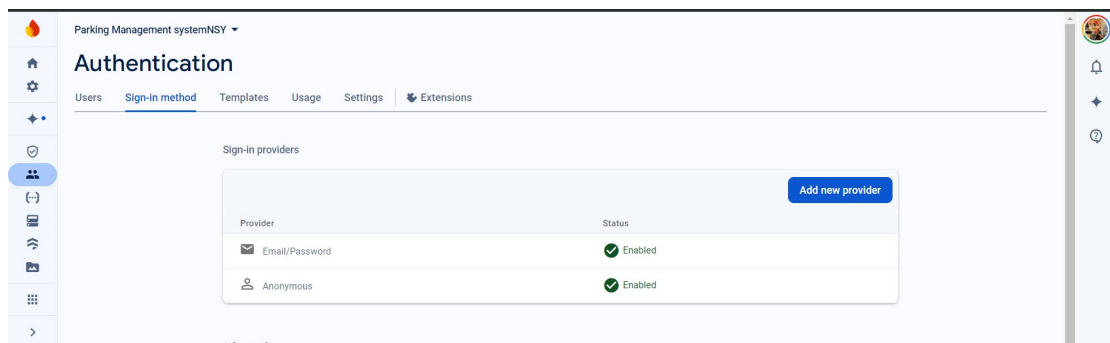


Figure 5.2.1.3 Enable Authentication Methods in Firebase Console.

Step 4: Use Firebase Authentication to handle user register and sign in for Android Application

The Register class is an Android activity that implements a user registration screen using Firebase Authentication SDK and Firestore. When the activity is created, it initializes the Firebase Authentication instance (`mAuth = FirebaseAuth.getInstance()`) to handle user account creation and the Firestore instance (`db = FirebaseFirestore.getInstance()`) to store additional user data such as role and email. The user interface includes input fields for email and password, a registration button, and a link to navigate to the login screen. When a user enters their details and clicks the register button, the `registerUser()` method is invoked. This method first validates that the email and password fields are not empty. If valid, it calls `createUserWithEmailAndPassword()` from the Firebase Authentication SDK to create a new user account.

Upon successful registration, the app retrieves the current `FirebaseUser` and assigns a role based on the email address—if the email is `management@gmail.com`, the user is given the “management” role; otherwise, the default role is “resident.” This role, along with the email, is stored in the Firestore database under a collection named "users" using the `.set()` method. If the data is successfully written to Firestore, the user is redirected to the login screen. If registration or Firestore saving fails at any point, appropriate error messages are displayed using Toast messages. This implementation showcases a typical use of the Firebase Authentication SDK to manage user sign-up securely, while Firestore is used to persist custom user attributes such as roles. Overall,

the activity ensures a smooth and secure onboarding process for users in a mobile application environment.

```
public class Register extends AppCompatActivity {

    private EditText emailEditText;
    private EditText passwordEditText;
    private TextView loginTextView;

    private FirebaseAuth mAuth;
    private FirebaseFirestore db;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);

        emailEditText = findViewById(R.id.emailEditText);
        passwordEditText = findViewById(R.id.passwordEditText);
        Button registerButton = findViewById(R.id.registerButton);
        loginTextView = findViewById(R.id.loginTextView);

        mAuth = FirebaseAuth.getInstance();
        db = FirebaseFirestore.getInstance();

        registerButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                registerUser();
            }
        });

        loginTextView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Open login activity
                startActivity(new Intent(Register.this, Login.class));
            }
        });
    }

    private void registerUser() {
        String email = emailEditText.getText().toString().trim();
        String password = passwordEditText.getText().toString().trim();

        if (TextUtils.isEmpty(email)) {
            emailEditText.setError("Email is required");
            return;
        }
    }
}
```

```

        if (TextUtils.isEmpty(password)) {
            passwordEditText.setError("Password is required");
            return;
        }

        mAuth.createUserWithEmailAndPassword(email, password)
            .addOnCompleteListener(this, task -> {
                if (task.isSuccessful()) {
                    // Registration success
                    FirebaseUser user = mAuth.getCurrentUser();
                    String role = email.equals("management@gmail.com") ?
"management" : "resident";

                    // Save role and email to Firestore
                    if (user != null) {
                        db.collection("users").document(user.getEmail()).set(new
User(email, role))
                            .addOnCompleteListener(dbTask -> {
                                if (dbTask.isSuccessful()) {
                                    // Navigate to login activity
                                    startActivity(new Intent(Register.this, Login.class));
                                    finish();
                                } else {
                                    Toast.makeText(Register.this, "Error saving user data: "
+ dbTask.getException().getMessage(), Toast.LENGTH_SHORT).show();
                                }
                            });
                    }
                } else {
                    // Registration failed
                    String errorMessage = task.getException() != null ?
task.getException().getMessage() : "Registration failed";
                    Toast.makeText(Register.this, "Registration failed: " + errorMessage,
Toast.LENGTH_SHORT).show();
                }
            });
    }
}

```

Table 5.2.1.1 Example to use Firebase Authentication to handle user register

2. Firebase Firestore

Step 1: Include the Firestore dependency in build.gradle (Module:App) file

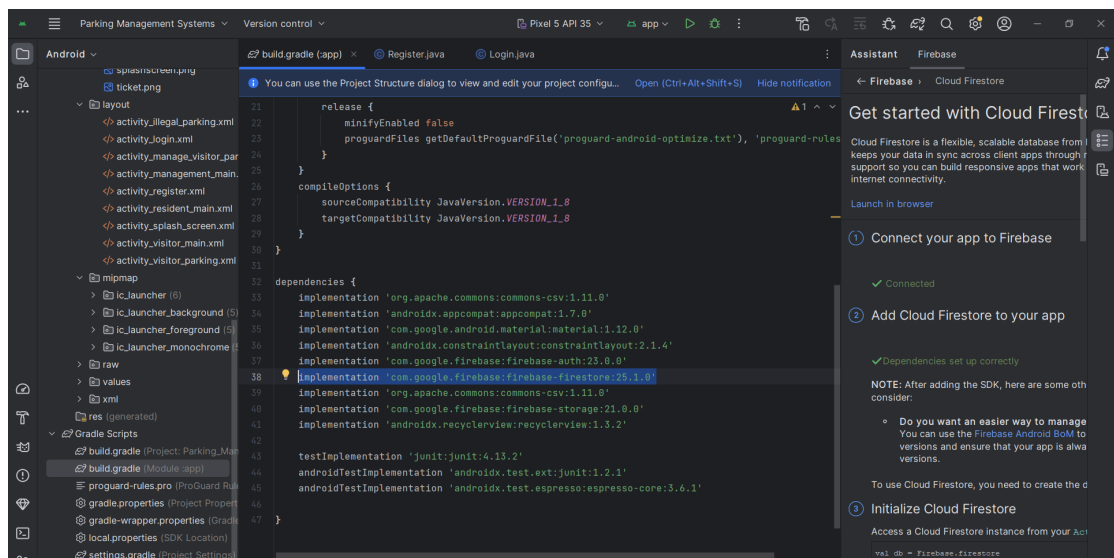


Figure 5.2.1.4 Add Firestore dependencies in build.gradle

Step 2: Sync project with Gradle files.

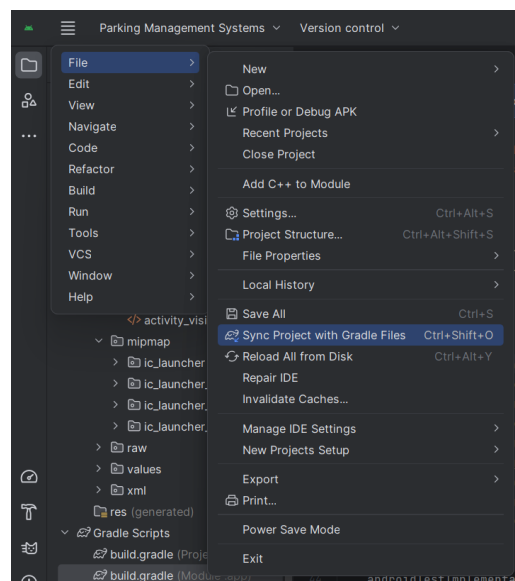


Figure 5.2.1.5 Sync the project with Gradle Files

Step 3: In the Firebase Console, set up your Firestore database and define the data structure and security rules.

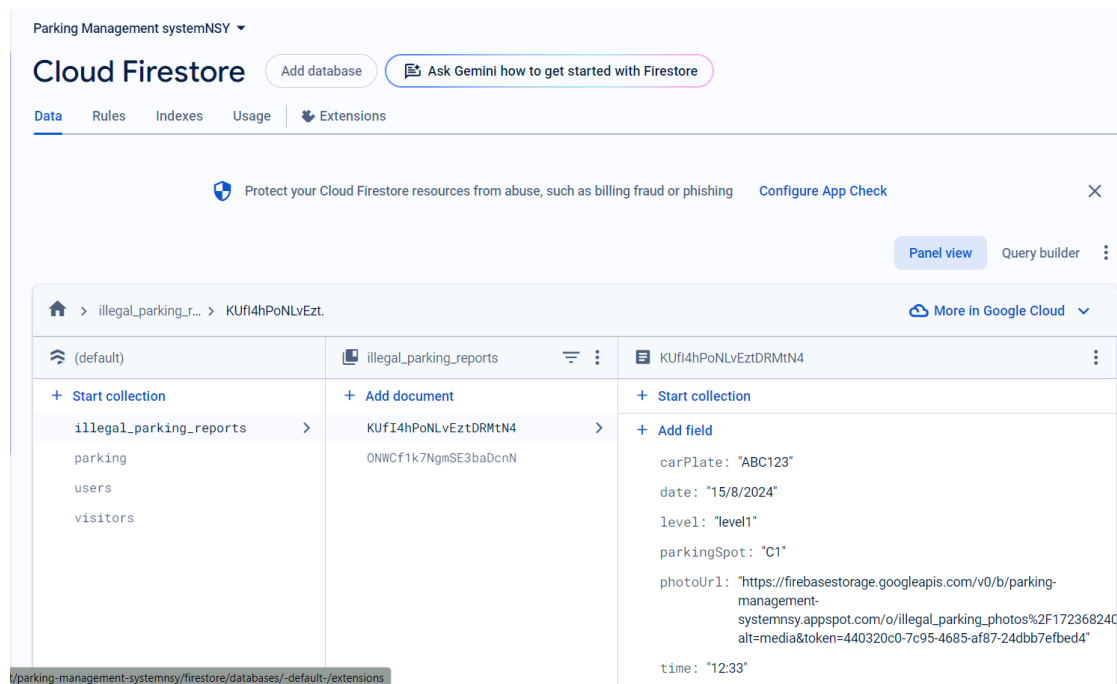


Figure 5.2.1.6 Set Up Firestore Database

Step 4: Use Firestore to read and write data in your Android code. The below example is to store the visitor details.

The VisitorMainActivity class represents the main interface for visitors within the parking management mobile application. In this activity, Firebase Firestore is initialized using `db = FirebaseFirestore.getInstance();`. Firestore is a cloud-hosted NoSQL database provided by Firebase that enables real-time data synchronization and flexible document storage. By initializing Firestore in this activity, the application can perform operations such as retrieving visitor parking data, validating reference codes, and checking parking availability. This setup allows the app to interact with cloud-based data efficiently, ensuring that visitor-related operations such as authorization, navigation, and parking assignment are always up to date. The use of Firestore provides scalability and reliability for managing dynamic parking data in real time, making it ideal for mobile applications where multiple users may interact with the system concurrently.

```
public class VisitorMainActivity extends AppCompatActivity {
    private FirebaseFirestore db;
```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_visitor_main);

    // Initialize Firestore
    db = FirebaseFirestore.getInstance();

public class VisitorMainActivity extends AppCompatActivity {

    private FirebaseFirestore db;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_visitor_main);

        // Initialize Firestore
        db = FirebaseFirestore.getInstance();

```

Table 5.2.1.2 Example to use Firestore to Store Visitor Details

3. Firebase Storage

Step 1: Include the Firesbase Storage dependency in your build.gradle (Module:App) file.

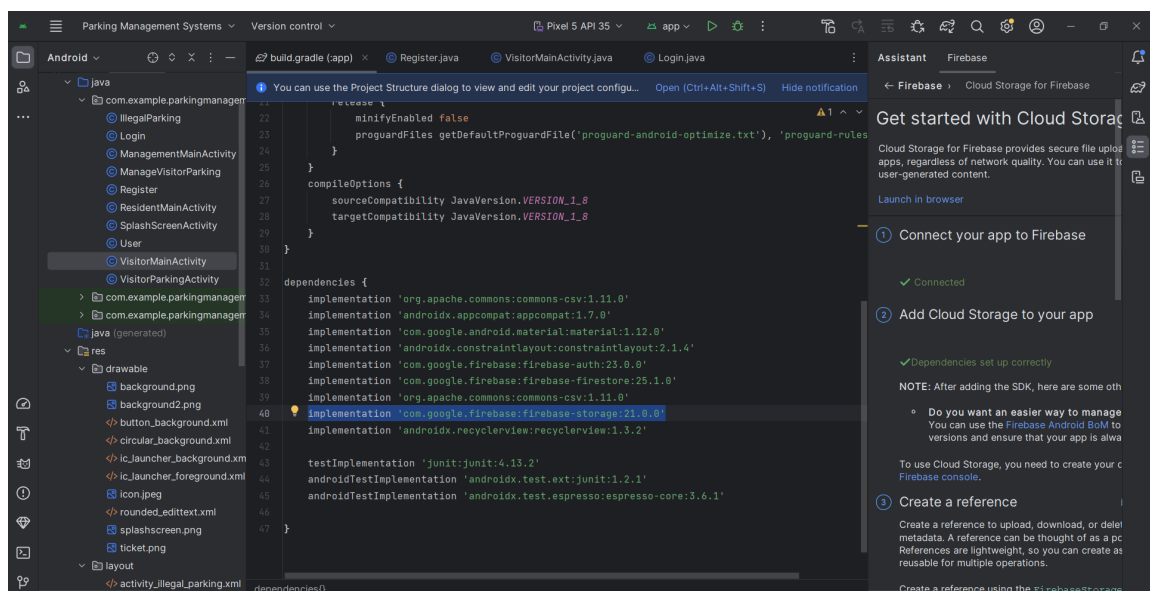


Figure 5.2.1.7 Add Firebase Storage SDK dependencies in build.gradle

Step 2: Sync project with Gradle files.

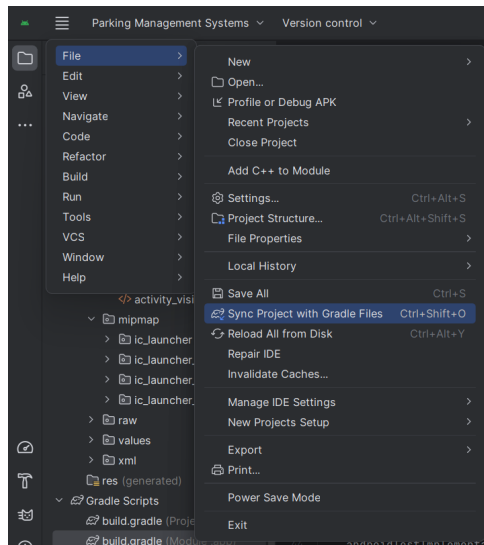


Figure 5.2.1.8 Sync project with Gradle Files

Step 3: In the Firebase Console, set up Firebase Storage and configure the security rules.

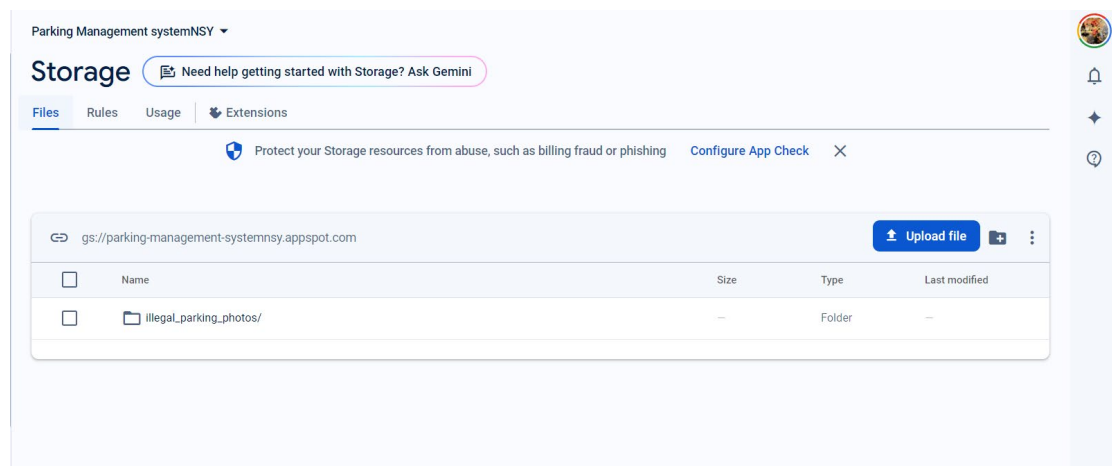


Figure 5.2.1.9 Set up Firebase Storage

Step 4: Use Firebase Storage to upload and download files. The below is an example of store the image of illegal parking into Firebase Storage.

In the illegal parking reporting feature, Firebase Storage is used to store photographic evidence of parking violations, while Firebase Firestore handles the associated textual data. The Firebase services are first initialized using `FirebaseFirestore.getInstance()` and `FirebaseStorage.getInstance()`. When a visitor or

resident submits a report, the image of the illegally parked vehicle is uploaded to Firebase Storage using a time-stamped path such as "illegal_parking_photos/{timestamp}.jpg", ensuring each file is uniquely identified. This is done via a `StorageReference` and the `putFile(photoUri)` method, which handles the actual file upload. Upon a successful upload, the app retrieves the image's public download URL using `getDownloadUrl()`.

This URL, along with other details such as the parking level, parking spot, date, time, and car plate number, is stored in a Firestore collection named "illegal_parking_reports" as a new document. The app uses a `HashMap` to structure this data and calls the `.add()` method to insert it into Firestore. This allows both the photo and report details to be securely stored and accessed by management in real time. Users receive feedback on the submission status via toast messages, confirming whether the report was submitted successfully or if an error occurred. The combination of Firebase Storage and Firestore ensures that multimedia evidence is handled efficiently while maintaining a scalable and cloud-based report management system.

```
db = FirebaseFirestore.getInstance();
storage = FirebaseStorage.getInstance();

StorageReference storageRef = storage.getReference("illegal_parking_photos/" +
System.currentTimeMillis() + ".jpg");
storageRef.putFile(photoUri)
    .addOnSuccessListener(taskSnapshot ->
storageRef.getDownloadUrl().addOnSuccessListener(uri -> {
    Map<String, Object> reportData = new HashMap<>();
    reportData.put("level", level);
    reportData.put("parkingSpot", parkingSpot);
    reportData.put("time", time);
    reportData.put("date", date);
    reportData.put("carPlate", carPlate);
    reportData.put("photoUrl", uri.toString());

    // Store the report in Firestore under the "illegal_parking_reports"
collection
    db.collection("illegal_parking_reports")
        .add(reportData)
        .addOnCompleteListener(new
OnCompleteListener<DocumentReference>() {
            @Override
```

```

        public void onComplete(@NonNull Task<DocumentReference>
task) {
            if (task.isSuccessful()) {
                Toast.makeText(IllegalParking.this, "Report submitted
successfully", Toast.LENGTH_SHORT).show();
                clearForm();
            } else {
                String errorMessage = task.getException() != null ?
task.getException().getMessage() : "Unknown error";
                Toast.makeText(IllegalParking.this, "Error submitting report:
" + errorMessage, Toast.LENGTH_LONG).show();
            }
        }
    });
}

    .addOnFailureListener(e -> Toast.makeText(IllegalParking.this, "Photo
upload failed: " + e.getMessage(), Toast.LENGTH_LONG).show());
}

```

Table 5.2.1.3 Example to use Firebase Storage to Store Illegal Parking Photo

5.3 System Operation

5.3.1 Login Module

Initially, users will enter the login. They need to log in to access their respective main activities. There are three role in this system, management, resident and visitor. Different roles will have different access levels and main activities. For Management users, they must use a management email to enter the application. Residents can log in using their registered email and password. Visitors only need to click on "I'm a Visitor" to access their main activity. They will be redirected to their respective main activities based on their role. If users do not have an account, they will need to register.

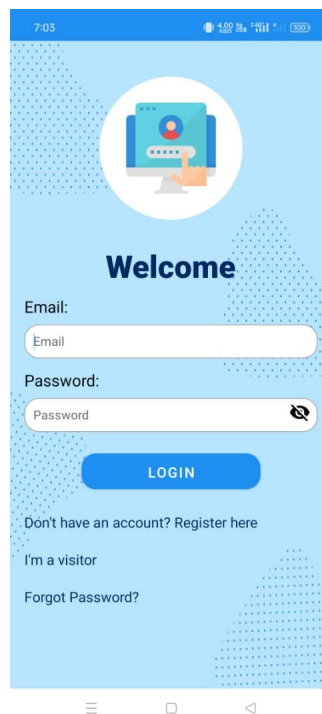


Figure 5.3.1 Login Module

5.3.2 Sign up module

Residents need to enter their username, email address, and password to register an account. If the user already has an account, they can proceed to log in. An email validation link will be sent to the user's email address for verification. The user must verify their email before they can proceed to Login.

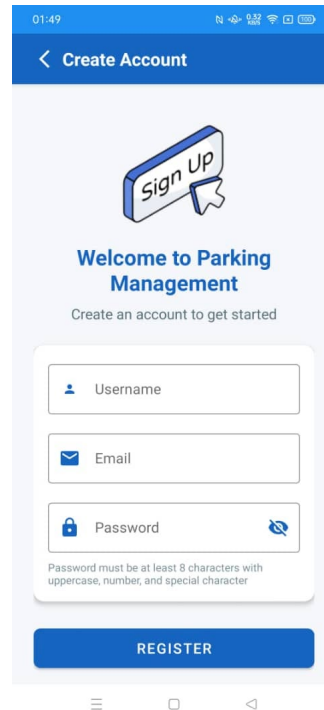
The image is a screenshot of a mobile application's 'Create Account' screen. At the top, there is a blue header bar with a white back arrow and the text 'Create Account'. Below the header, there is a light blue background area. In the center, there is a speech bubble icon with the text 'Sign Up' inside. Below the icon, the text 'Welcome to Parking Management' is displayed in a bold, dark blue font. Underneath this, in a smaller, lighter blue font, it says 'Create an account to get started'. Below this text, there are three input fields: 'Username' with a person icon, 'Email' with an envelope icon, and 'Password' with a lock icon and a toggle for visibility. Below the password field, there is a small text note: 'Password must be at least 8 characters with uppercase, number, and special character'. At the bottom of the form area, there is a large blue button with the text 'REGISTER' in white. The entire screen is framed by a white border, and at the very bottom, there are three small navigation icons: a hamburger menu, a square, and a back arrow.

Figure 5.3.2 Sign Up Module

5.3.3 Forgot Password

User needs to input their registered email address. After press on the “reset password” button, a reset password email will be sent to the user email address. By clicking the link user can reset the password and proceed to sign in.

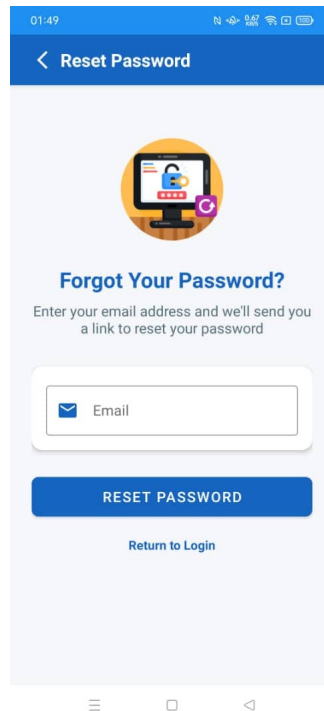


Figure 5.3.3 Forgot Password Module

5.3.4 Visitor Module

The Visitor Module is designed to streamline the process of visitor parking management while ensuring that only authorized visitors are granted access to the premises. In the current implementation, the module operates through a reference code system that links visitor access to resident approval.

Rather than allowing visitors to self-register, the process begins with the resident registering the visitor's details through the system. Upon successful registration, the system generates a unique reference code, which the resident then provides to the visitor. This code acts as a security measure to verify that the visitor is genuinely invited.

When the visitor arrives, they are required to input the reference code into the mobile application. The system then verifies the code against records stored in the database. If the code is valid and has not been previously used, the system automatically assigns an available parking space to the visitor. This assignment is handled through the existing auto-assign carpark feature, which checks for available spots across multiple levels and assigns one accordingly.

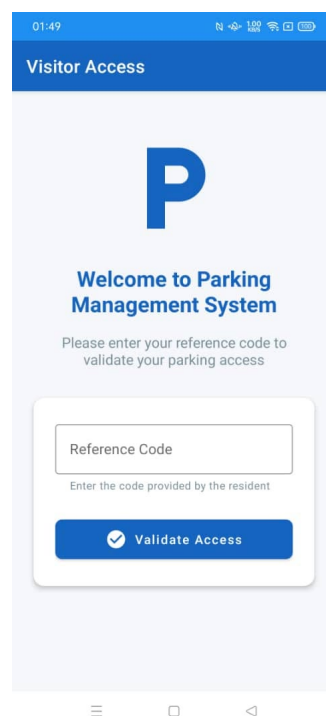


Figure 5.3.4 Visitor Module

5.3.5 Carpark Spot Auto Assignment

After submitting the details, the system will automatically assign a parking space for the visitor. It will check Firestore for the latest available parking spots to ensure there are no conflicts with other visitors using the same space. After the assignment, the system should display a floor plan to guide the visitor to the parking space.

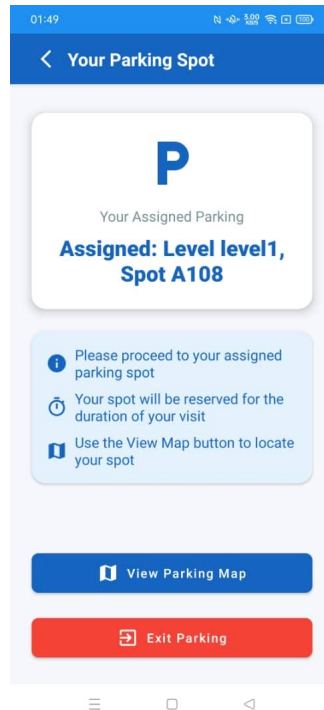


Figure 5.3.5 Carpark Spot Auto Assignment Module

5.3.6 View Map Module

After a parking slot has been auto assigned to the visitor, the system provides a "View Map" feature to guide the user to the allocated spot. This module displays a visual floor plan of the parking layout, showing the entrance, exit, and directional arrows for navigation. The assigned parking lot is clearly highlighted, helping users locate their spot easily without confusion. This improves the visitor experience by providing intuitive and direct visual guidance upon arrival.

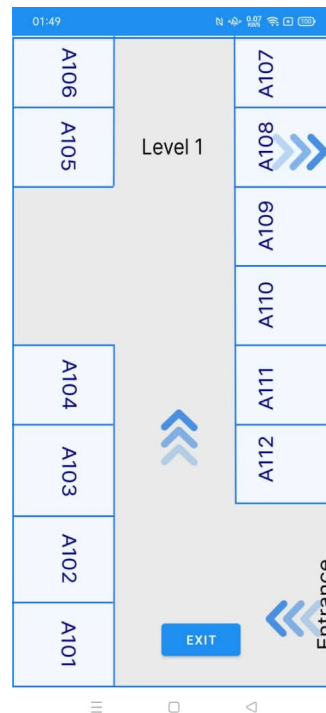


Figure 5.3.6 View Map Module

5.3.7 Resident Main Module

The Resident Main Module serves as the central interface for residents to access key features within the application. It includes three main services: Report Illegal Parking, which allows residents to report unauthorized vehicles by submitting details and an evidence photo; Register Visitor, where residents can register incoming guests and generate a reference code for visitor parking; and View My Complaints, which enables residents to track the status of their submitted illegal parking reports. At the bottom of the page, there is a Logout button under the Account section, allowing users to securely sign out of the application. The layout is designed to be user-friendly and ensures residents can easily manage their parking-related tasks.

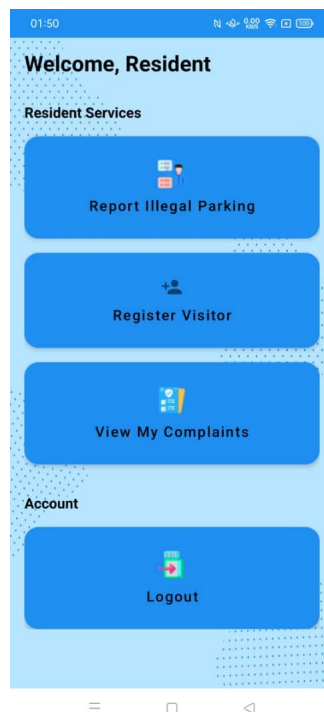
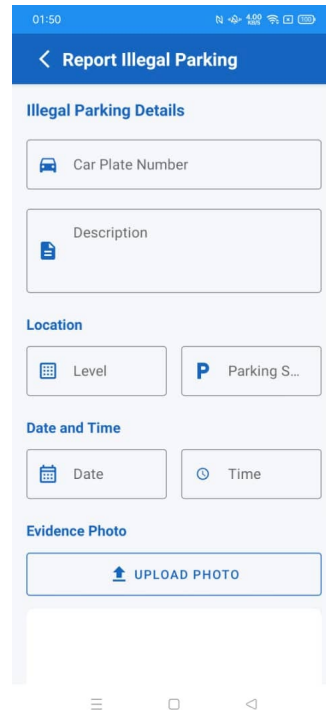


Figure 5.3.7 Resident Main Module

5.3.8 Report Illegal Parking Module

Residents can report illegal parking by filling in the details of the offending vehicle and uploading an evidence photo. The report will be received by management immediately. After submitting the illegal parking report, the resident will be asked if they need a parking space. If yes, the system will automatically assign a temporary parking space.



The screenshot displays a mobile application interface for reporting illegal parking. At the top, a blue header bar contains a back arrow and the title 'Report Illegal Parking'. Below this, the form is organized into several sections: 'Illegal Parking Details' with input fields for 'Car Plate Number' and 'Description'; 'Location' with fields for 'Level' and 'Parking S...'; 'Date and Time' with fields for 'Date' and 'Time'; and 'Evidence Photo' with an 'UPLOAD PHOTO' button. The interface is clean and uses a light blue color scheme. At the bottom, there are three small navigation icons.

Figure 5.3.8 Report Illegal Parking Module

5.3.9 Register Visitor Module

The Register Visitor Module allows residents to register their visitors in advance by entering key information such as the visitor's full name, ID card number, car plate number, and phone number. The form ensures that all required details are captured accurately to facilitate proper identification. Once the registration is completed, the system will automatically proceed to generate a unique reference code linked to the visitor's record. This reference code must be shared with the visitor, who will use it to verify their entry and receive an assigned parking space upon arrival. This process enhances security and ensures that only authorized visitors are granted access to the parking facilities.

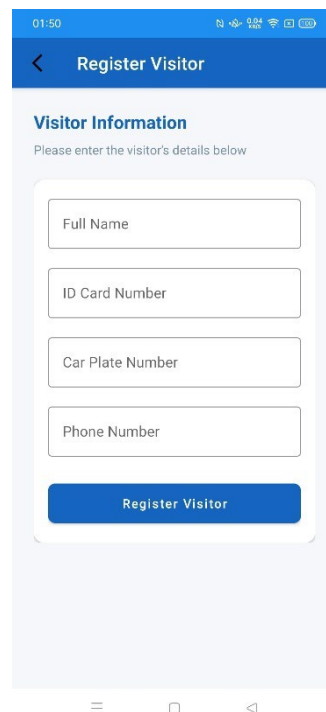
The image shows a mobile application interface for the 'Register Visitor' module. At the top, there is a blue header bar with a back arrow and the text 'Register Visitor'. Below this, the section is titled 'Visitor Information' with a subtitle 'Please enter the visitor's details below'. The form contains four input fields: 'Full Name', 'ID Card Number', 'Car Plate Number', and 'Phone Number'. At the bottom of the form is a blue button labeled 'Register Visitor'. The entire form is set against a light blue background. The mobile status bar at the very top shows the time as 01:50 and various icons for signal, Wi-Fi, and battery.

Figure 5.3.9 Register Visitor Module

5.3.10 Visitor Reference Code Module

After a visitor is registered, the system generates a unique reference code through the Visitor Reference Code Module. This code acts as a secure key to verify the visitor's identity upon arrival. The interface displays the code clearly and provides two options: Copy, which allows the resident to easily share the code with the visitor, and Done, to exit the screen. The visitor must enter this code in the application to validate their entry and receive an auto-assigned parking space. This module enhances both security and convenience by ensuring that only visitors approved by residents can access the parking facilities.

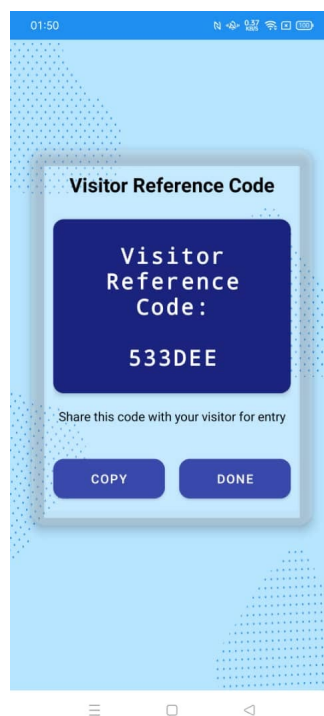


Figure 5.3.10 Visitor Reference Code Module

5.3.11 View My Reports Module

The View My Reports Module allows residents to view the status and history of the illegal parking reports they have submitted. The interface provides a summary count of reports under three categories: Pending, In Progress, and Resolved. Each report is listed with key details such as the car plate number, location, description, timestamp, and an evidence photo. The current status is clearly labelled for each report, helping residents track whether action has been taken. This module enhances transparency and ensures residents stay informed about the outcome of their submitted complaints.

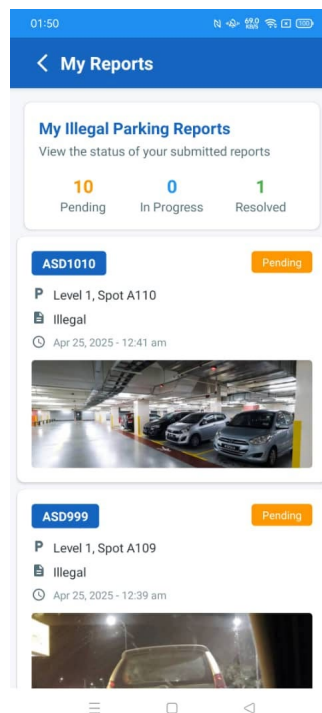


Figure 5.3.11 View My Reports Module

5.3.12 Management Main Page

The Management Main Page serves as the central hub for all administrative functions within the application. It provides easy access to essential features through a clean and organized interface. Under Management Services, staff can access the Illegal Parking Report module to handle reports submitted by residents and the Manage Visitor Parking module to update visitor parking availability.

In the NFC Management section, management can perform NFC-related tasks, including writing data to NFC tags and using the Read-Only NFC Scan feature to verify vehicle tags without making changes.

At the bottom of the page, there is a Logout button that allows the user to securely exit the management account.

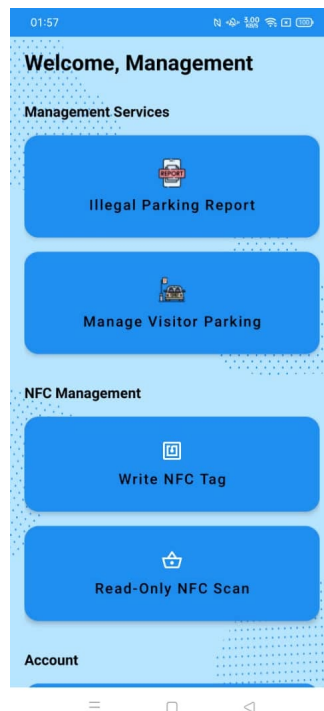


Figure 5.3.12 Management Main Module

5.3.13 Manage Visitor Parking Module

The Manage Visitor Parking Module allows management to update the availability of visitor parking spaces, especially when certain spots are rented out and need to be marked as unavailable. To support different update needs, the system offers two options: uploading a CSV file for bulk changes or using an interactive floor plan for quick updates. The floor plan uses colour coding green for available, red for occupied, and blue for selected. To make the status of each parking spot clear and easy to manage. This ensures accurate and efficient parking allocation.

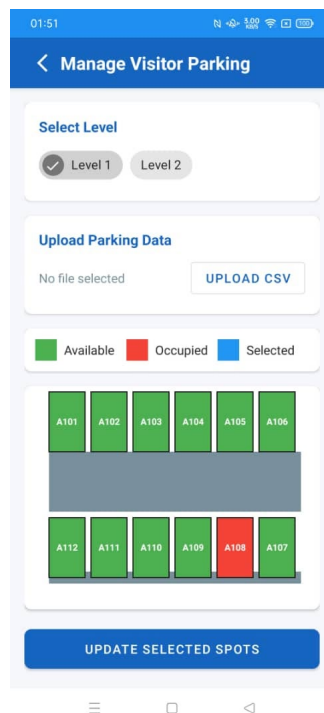


Figure 5.3.13 Manage Visitor Parking Module

5.3.14 Illegal Parking Report Module

This module is designed for the management team to view and handle illegal parking reports submitted by residents. All reports are retrieved in real-time from Firebase and displayed using a structured card view layout, making the information clear and easy to read. Each report includes essential details such as the vehicle's plate number, location, timestamp, description, and an attached photo. To ensure efficient case management, the system allows management to update the status of each report using three predefined options: *Pending*, *In Progress*, and *Resolved*. This status-tracking system promotes accountability and provides transparency to both management and residents. Additionally, a push notification feature has been implemented. When a resident submits an illegal parking report, a notification is instantly sent to the management's mobile device. This enables prompt response and faster handling of reported cases

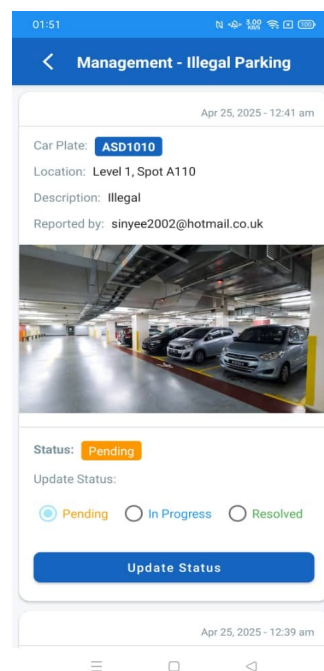


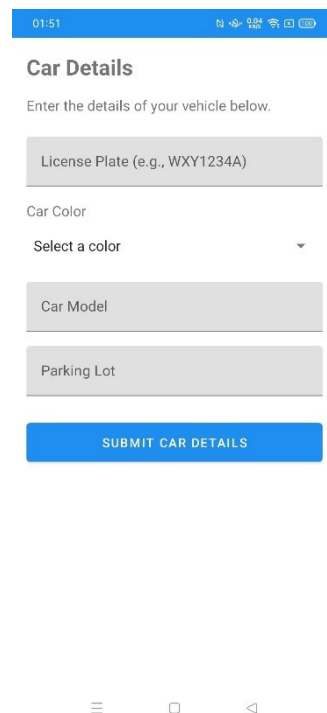
Figure 5.3.14.1 Illegal Parking Report



Figure 5.3.14.2 Illegal Parking Notification

5.3.15 NFC Car Details Module

The Car Details Module is used by management to input the vehicle information of a resident before writing the data onto an NFC tag. The form collects essential details such as the car's license plate number, color, model, and the assigned parking lot. Once the form is completed and submitted, the system proceeds to the Write NFC Tag module, where this information is encoded into the tag. This ensures that each NFC tag securely stores verified vehicle and parking details, allowing for quick and secure identification when scanned.



The screenshot shows a mobile application interface for the 'Car Details' module. At the top, a blue status bar displays the time '01:51' and various system icons. Below this, the title 'Car Details' is centered in bold. A subtitle 'Enter the details of your vehicle below.' is positioned just above the input fields. The form consists of four stacked text input fields: 'License Plate (e.g., WXY1234A)', 'Car Color' (with a dropdown arrow), 'Car Model', and 'Parking Lot'. At the bottom of the form is a prominent blue button labeled 'SUBMIT CAR DETAILS'. Below the form, there are three small, faint navigation icons: a hamburger menu, a square, and a back arrow.

Figure 5.3.15 NFC Car Details Module

5.3.16 NFC Tag Writer Module

The NFC Tag Writer Module is used to write resident vehicle information onto an NFC tag. After submitting the car details, this screen prompts the user to place the NFC tag near the mobile device to begin the writing process. The displayed car details such as plate number, colour, model, and assigned parking lot are encoded directly into the tag. To proceed, the user must ensure that NFC is turned on and permissions are granted. This step is essential for enabling secure, contactless identification of vehicles within the parking management system. Once the writing process is successful, the interface provides clear feedback by turning green, indicating that the tag has been successfully written and is ready for use.

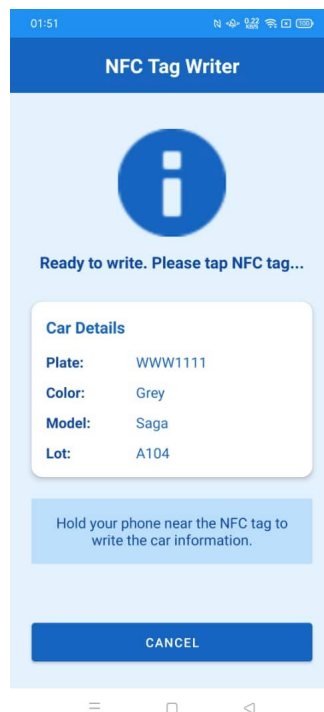


Figure 5.3.16.1 NFC Tag Writer Module

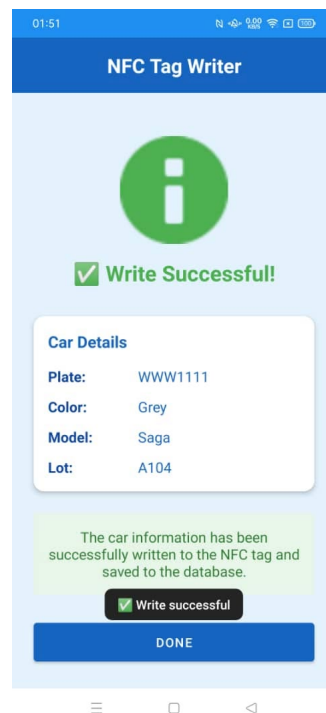


Figure 5.3.16.2 NFC Tag Write Successfully

5.3.17 NFC Tag Reader

The NFC Tag Reader Module is designed for management to quickly verify car details by scanning an NFC tag. When the screen is active, the user is prompted to hold the phone near the NFC tag, which triggers the reading process. Once the tag is detected, the stored information—such as car plate number, colour, model, and parking lot—will be displayed on the screen. This module requires NFC to be enabled and appropriate permissions granted to function correctly. It provides a fast and secure way for management to confirm registered vehicles within the parking system.

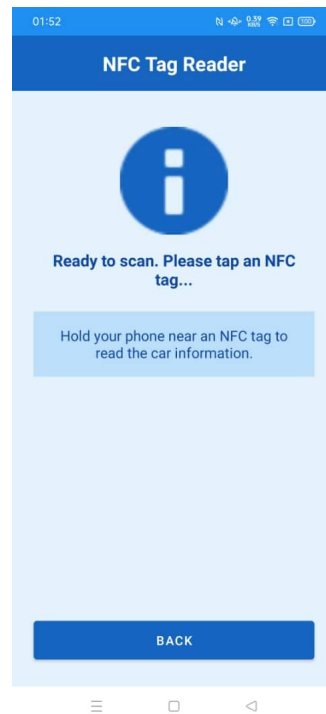


Figure 5.3.17.1 NFC Tag Reader Module

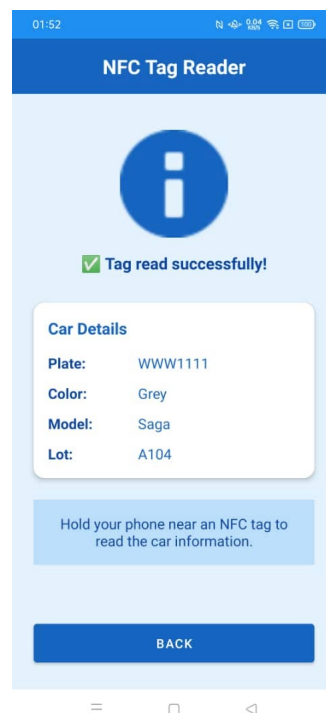


Figure 5.3.17.2 NFC Tag Reader Successfully

5.4 Concluding Remark

The implementation of the Parking Management System with Mobile NFC Authentication marks a major step in turning the project plan into a working mobile application. Using Android Studio for development and Firebase as the backend, the system combines secure authentication, real-time data management, and contactless NFC technology to handle parking operations more efficiently.

Throughout this phase, each feature was carefully developed to meet the project's goals—such as automatically assigning parking spots, managing visitor access through reference codes, reporting illegal parking, and verifying car details using NFC tags. The system makes use of Firebase Authentication to handle user login securely, Firestore to store and retrieve real-time data, and Firebase Storage for managing uploaded photos, like illegal parking evidence.

One of the key highlights was the NFC functionality, which allows management to write car details to an NFC tag and later read them back for verification. This not only improves security but also reduces manual paperwork and speeds up the inspection process. Another important feature is the Manage Visitor Parking module, where management can either upload a CSV file for bulk updates or interact with a floor plan to mark spot availability—making the process both flexible and practical.

Overall, the system has achieved what it set out to do. The application is fully functional, user-friendly, and meets the core objectives set at the beginning of the project. All the main modules were implemented and tested successfully, and the features are ready to be used in real-world scenarios. This lays a strong foundation for the next phase of development, where more advanced enhancements can be added based on feedback and future needs.

CHAPTER 6

System Evaluation and Discussion

6.1 System Performance Evaluation

6.1.1 Loading Test

To evaluate the image rendering performance in the Parking Management System with Mobile NFC Authentication, a loading test was conducted on the View My Reports module. This module allows residents to view their submitted illegal parking reports, each of which contains a photo stored in Firebase Cloud Storage. Since images are an essential part of report verification and the RecyclerView displays multiple items with images, it is critical to ensure that the loading experience remains fast and responsive.

The test aimed to measure the time it takes to load each report's image and render it on the screen. To achieve this, the IllegalParkingReportAdapter was instrumented using Glide's RequestListener to log the load start and end times for each image. The difference in milliseconds was then recorded using System.currentTimeMillis() to calculate the duration taken to load each image. These logs were printed to Logcat with the report ID and the corresponding load time.

```
00:17:03.456 I Image loaded for report 5S8SbwjjTAWvi7ssjMLD in 3 ms
00:17:03.701 I Image loaded for report PqqIskoLEwGsrhUU7tqj in 266 ms
00:17:08.267 I Image loaded for report UnUM4yRDtLtdU6VZqj14 in 50 ms
00:17:08.691 I Image loaded for report 0fWlfbxVxDTRpGoMxTex in 393 ms
00:17:08.882 I Image loaded for report CUbtP7MNJUfNwasTJUIW in 404 ms
00:17:08.942 I Image loaded for report cTePfmq000EghvVuwQCB in 259 ms
00:17:09.059 I Image loaded for report LfpetBK64WeXeIgT9VZi in 2 ms
00:17:09.079 I Image loaded for report TsuySMFtJ9HC2eQw8mSZ in 336 ms
00:17:09.176 I Image loaded for report b6Nb4zjEyPCViamYX0kj in 350 ms
00:17:09.218 I Image loaded for report Ke98iySQq8z9tz4TMW0X in 294 ms
```

Figure 6.1.1 Loading Test for report image

A total of **10 reports** were used for this test, and their images were loaded sequentially during the activity lifecycle. The recorded load times ranged from **2 ms to 404 ms**, with the majority of images loading in under 350 ms. This demonstrates that the system can handle image rendering efficiently under typical usage scenarios, maintaining good user experience without noticeable delays or UI lag.

This test confirms that the application performs well in loading remote images from Firebase storage within acceptable performance bounds. The approach used also provides a reusable method for future performance tracking and optimization.

6.1.2 Response Time Analysis

The Parking Management System with Mobile NFC Authentication incorporates an auto-assign module designed to dynamically allocate the nearest available parking spot to visitors. This module is essential in ensuring a seamless and efficient parking experience, particularly in high-traffic residential areas. To evaluate its performance, a response time analysis was conducted to measure how long the system takes to assign a parking spot or determine unavailability.

The auto-assign logic sequentially checks three parking levels—Level 1 and Level 2 by querying the Firestore database for available lots. When a spot is found, the system updates its status in Firestore and displays the assignment to the user. The system records the start and end times using `System.currentTimeMillis()` in Java, allowing the total duration of the assignment process to be calculated and logged. This provides an accurate measure of system responsiveness from the moment the check begins until the assignment is confirmed or all levels have been exhausted.

| | | |
|--------------|---|--|
| 23:47:15.423 | I | Auto-assign result: Assigned at Level 1 Response time: 314 ms |
| 23:47:44.462 | I | Auto-assign result: Assigned at Level 1 Response time: 237 ms |
| 23:48:31.920 | I | Auto-assign result: Assigned at Level 1 Response time: 938 ms |
| 23:49:00.194 | I | Auto-assign result: Assigned at Level 1 Response time: 296 ms |
| 23:49:26.184 | I | Auto-assign result: Assigned at Level 1 Response time: 172 ms |
| 23:50:07.446 | I | Auto-assign result: Assigned at Level 2 Response time: 263 ms |
| 23:50:42.349 | I | Auto-assign result: Assigned at Level 2 Response time: 342 ms |
| 23:51:12.721 | I | Auto-assign result: Assigned at Level 2 Response time: 215 ms |
| 23:51:42.254 | I | Auto-assign result: Assigned at Level 2 Response time: 321 ms |
| 23:52:12.240 | I | Auto-assign result: Assigned at Level 2 Response time: 270 ms |
| 23:52:42.575 | I | Auto-assign result: No spot found in any level Response time: 258 ms |
| 23:53:11.918 | I | Auto-assign result: No spot found in any level Response time: 230 ms |
| 23:53:40.128 | I | Auto-assign result: No spot found in any level Response time: 185 ms |
| 23:54:15.583 | I | Auto-assign result: No spot found in any level Response time: 336 ms |
| 23:54:47.102 | I | Auto-assign result: No spot found in any level Response time: 190 ms |

Figure 6.1.2 Response Time Analysis for Auto Assignment Parking

Testing was conducted a total of 15 times to evaluate the consistency and reliability of the response time. Specifically, five test runs were completed where parking spots were successfully assigned at Level 1, five runs at Level 2, and another five where no parking spots were available across all levels. Successful assignments at Level 1 recorded response times ranging between 172 ms and 938 ms, while Level 2 assignments averaged between 215 ms and 342 ms. In cases where no available parking spots were found, the response time ranged from 185 ms to 336 ms. These results demonstrate that the module is highly responsive, with the majority of operations completing in under 400 milliseconds. This level of responsiveness is well within acceptable limits for real-time mobile applications and confirms the system's suitability for practical use in a smart parking environment.

6.1.3 Network Performance

The Illegal Parking feature of the Parking Management System allows residents to report unauthorized vehicle parking by uploading photographic evidence to Firebase Storage. To evaluate the system's responsiveness and efficiency, a network performance test was conducted focusing on the image upload process. The test aimed to determine how long the application takes to upload images under standard Wi-Fi conditions using Android's `System.currentTimeMillis()` to log the upload start and end times.

| | | |
|--------------|---|--------------------------|
| 00:32:57.214 | I | Image uploaded in 977 ms |
| 00:33:48.486 | I | Image uploaded in 592 ms |
| 00:34:52.055 | I | Image uploaded in 748 ms |
| 00:35:44.429 | I | Image uploaded in 685 ms |
| 00:36:26.055 | I | Image uploaded in 593 ms |
| 00:37:07.124 | I | Image uploaded in 846 ms |
| 00:37:50.693 | I | Image uploaded in 824 ms |
| 00:39:13.733 | I | Image uploaded in 715 ms |
| 00:39:55.954 | I | Image uploaded in 515 ms |
| 00:41:11.436 | I | Image uploaded in 830 ms |

Figure 6.1.3. Network Performance for Image Upload to Database

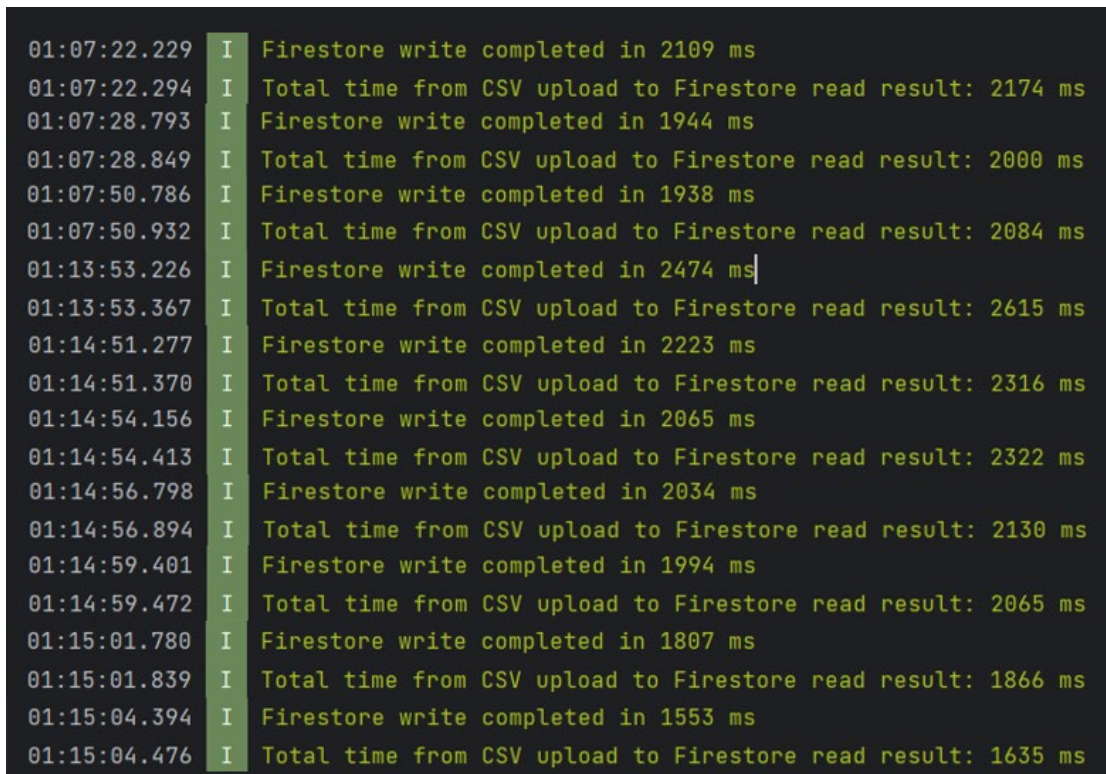
The test was performed by submitting 10 different illegal parking reports, each with a distinct image ranging in file size from approximately 500 KB to 2 MB. The results were recorded through Logcat and showed that the image upload time ranged from 515 ms to 977 ms, with most uploads completing within 600–850 ms. The average upload time across the 10 submissions was approximately 723 ms, which falls within acceptable limits for real-time mobile interaction.

6.1.4 Database Performance

To evaluate the database performance of the Parking Management System with Mobile NFC Authentication, a test was conducted to measure the duration required for Firestore to process bulk updates and return updated results after a CSV upload by the management. The test was performed through the ManageVisitorParking module, which allows administrators to upload a CSV file that updates the availability status of parking spots

The performance test involved recording two key timestamps using `System.currentTimeMillis()`:

- The duration from the start of the Firestore write operation to its completion.
- The total round-trip time from the beginning of the CSV upload to the point when Firestore returns the updated data through a `get()` query.



| | | |
|--------------|---|--|
| 01:07:22.229 | I | Firestore write completed in 2109 ms |
| 01:07:22.294 | I | Total time from CSV upload to Firestore read result: 2174 ms |
| 01:07:28.793 | I | Firestore write completed in 1944 ms |
| 01:07:28.849 | I | Total time from CSV upload to Firestore read result: 2000 ms |
| 01:07:50.786 | I | Firestore write completed in 1938 ms |
| 01:07:50.932 | I | Total time from CSV upload to Firestore read result: 2084 ms |
| 01:13:53.226 | I | Firestore write completed in 2474 ms |
| 01:13:53.367 | I | Total time from CSV upload to Firestore read result: 2615 ms |
| 01:14:51.277 | I | Firestore write completed in 2223 ms |
| 01:14:51.370 | I | Total time from CSV upload to Firestore read result: 2316 ms |
| 01:14:54.156 | I | Firestore write completed in 2065 ms |
| 01:14:54.413 | I | Total time from CSV upload to Firestore read result: 2322 ms |
| 01:14:56.798 | I | Firestore write completed in 2034 ms |
| 01:14:56.894 | I | Total time from CSV upload to Firestore read result: 2130 ms |
| 01:14:59.401 | I | Firestore write completed in 1994 ms |
| 01:14:59.472 | I | Total time from CSV upload to Firestore read result: 2065 ms |
| 01:15:01.780 | I | Firestore write completed in 1807 ms |
| 01:15:01.839 | I | Total time from CSV upload to Firestore read result: 1866 ms |
| 01:15:04.394 | I | Firestore write completed in 1553 ms |
| 01:15:04.476 | I | Total time from CSV upload to Firestore read result: 1635 ms |

Figure 6.1.4 Database Performance for CSV file updated the Database

The test was repeated 10 times, and the logged results indicate that Firestore write durations ranged from 1,553 ms to 2,474 ms, while the total time from CSV upload to Firestore read result ranged from 1,635 ms to 2,615 ms. On average, the write operation took approximately 2,016 ms, and the full round-trip including the read took approximately 2,183 ms.

These results demonstrate that Firestore can handle moderate-sized batch updates reliably and return the updated data within 2 to 3 seconds. This performance is considered acceptable for administrative tasks and batch updates that are not user-facing in real-time. The efficient performance of Firestore ensures that updated parking

spot availability can be accurately reflected in the system without noticeable delay to management or residents.

6.2 System Setting, Setup and Result

| Sign Up Test Case | | | | |
|---------------------------------------|--|--|---|------------|
| Use Case: Sign Up | | | | |
| Functional ID: F001 | | | | |
| Date Created: 24/4/2025 | | | | |
| Role: User | | | | |
| Test Case | Test Description | Input | Expected Output | Result |
| TC01 – Valid Registration | Register with valid username, email, and password | Valid username, valid email, strong password | Show success message and send verification email | 10/10 Pass |
| TC02 – Missing Fields | Attempt to register with missing inputs | Blank username or password | Show error message: 'Please fill in all fields' | 10/10 Pass |
| TC03 – Invalid Email Format | Enter incorrectly formatted email address | useremail.com | Show error: 'Invalid email format' | 10/10 Pass |
| TC04- Duplicate username | Try to register with an already used username | Previously registered username | Show error: 'Username is already Taken' | 10/10 Pass |
| TC04 – Weak Password | Use a password that doesn't meet security criteria | Password without numbers/symbols | Show error: 'Password must be at least 8 characters with a mix of characters' | 10/10 Pass |
| TC05 – Duplicate Email | Try to register with an already used email | Previously registered email | Show error: 'Email already in use' | 10/10 Pass |
| TC06 – Unverified Email Login Attempt | Try to log in without verifying the email | Valid sign-up, unverified email | Show error: 'Please verify your email before logging in' | 10/10 Pass |

Table 6.2.1 Sign Up Test Case

| Login Test Case | | | | |
|-------------------------------|--|--|--|------------|
| Use Case: Login | | | | |
| Functional ID: F002 | | | | |
| Date Created: 24/4/2025 | | | | |
| Role: User | | | | |
| Test Case | Test Description | Input | Expected Output | Result |
| TC01 – Valid Resident Login | Test resident login with valid credentials | Valid email and correct password | Redirect to Resident Main Page | 10/10 Pass |
| TC02 – Valid Management Login | Test management login with valid admin email | Valid admin email and correct password | Redirect to Management Main Page | 10/10 Pass |
| TC03 – Visitor Access | Test visitor login without credentials | Click "I'm a Visitor" | Redirect to Visitor Main Page | 10/10 Pass |
| TC04 – Invalid Email Format | Enter email without proper format | "useremail.com", "user@" | Show error message: "Invalid email format" | 10/10 Pass |
| TC05 – Wrong Password | Enter correct email with wrong password | Valid email, wrong password | Show error message: "Authentication Failed" | 10/10 Pass |
| TC06 – Empty Fields | Attempt to login with blank email and/or password fields | Empty input | Show error message: "Email and Password are required" | 10/10 Pass |
| TC07 – Unverified Email | Login with a registered email that is not verified | Registered but unverified email | Show error: "Please verify your email before logging in" | 10/10 Pass |

Table 6.2.2 Login Test Case

| Forgot Password Test Case | | | | |
|-----------------------------|---|--------------------|--|------------|
| Use Case: Forgot Password | | | | |
| Functional ID: F003 | | | | |
| Date Created: 24/4/2025 | | | | |
| Role: User | | | | |
| Test Case | Test Description | Input | Expected Output | Result |
| TC01 – Valid Email | Reset password with a registered email | Registered email | Display confirmation message and send reset link | 10/10 Pass |
| TC02 – Unregistered Email | Attempt to reset password with an email not in the system | Unregistered email | Show error: 'Email not found' | 10/10 Pass |
| TC03 – Empty Field | Submit form with no email entered | Empty input | Show error: 'Please enter your email' | 10/10 Pass |
| TC04 – Invalid Email Format | Submit form with incorrectly formatted email | useremail.com | Show error: 'Invalid email format' | 10/10 Pass |
| TC05 – Check Email Content | Ensure reset email contains valid reset link | Registered email | Email received with clickable reset link | 10/10 Pass |

Table 6.2.3 Forgot Password Test Case

| Visitor Module Test Case | | | | |
|--|---|--------------|--|------------|
| Use Case: Visitor Reference Code Entry | | | | |
| Functional ID: F004 | | | | |
| Date Created: 24/4/2025 | | | | |
| Role: Visitor | | | | |
| Test Case | Test Description | Input | Expected Output | Result |
| TC01 – Valid Reference Code | Enter a correct and unused reference code | Valid code | Assign parking spot and redirect to map view | 10/10 Pass |
| TC02 – Invalid Code | Enter a code that doesn't exist | Invalid code | Show error: 'Invalid reference code' | 10/10 Pass |
| TC03 – Empty Input | Submit without entering a code | Empty field | Show error: 'Please enter a | 10/10 Pass |

| | | | | |
|--|--|--|--------------------|--|
| | | | reference code' | |
|--|--|--|--------------------|--|

Table 6.2.4 Visitor Module Test Case

| Parking Assignment Test Case | | | | |
|---|---|-------------------------|---|------------|
| Use Case: Parking Assignment and View Map | | | | |
| Functional ID: F005 | | | | |
| Date Created: 24/4/2025 | | | | |
| Role: Visitor | | | | |
| Test Case | Test Description | Input | Expected Output | Result |
| TC01 – Assign from Level 1 | Assign parking when Level 1 has available spots | Trigger assignment | Assigned spot from Level 1 is returned | 10/10 Pass |
| TC02 – Fallback to Level 2 | Level 1 full, assign from Level 2 | Trigger assignment | Assigned spot from Level 2 is returned | 10/10 Pass |
| TC03 – All Levels Full | No spots available in any level | Trigger assignment | Show message: 'No available parking spots' | 10/10 Pass |
| TC04 – Assigned Spot Remains | Re-open application after assignment | Same visitor session | Previously assigned spot is shown | 10/10 Pass |
| TC05 – View Map Display | Display floor plan after assignment | Assigned spot and level | Show floor plan with entrance, direction, and assigned spot | 10/10 Pass |
| TC06 – Back Navigation | Click back from map view | Back button tapped | Return to previous screen with assignment retained | 10/10 Pass |

Table 6.2.5 Parking Assignment Test Case

| Report Illegal Parking Test Case | | | | |
|----------------------------------|--|--|--|------------|
| Use Case: Report Illegal Parking | | | | |
| Functional ID: F006 | | | | |
| Date Created: 24/4/2025 | | | | |
| Role: Resident | | | | |
| Test Case | Test Description | Input | Expected Output | Result |
| TC01 – Complete Report | Submit report with all required fields and a photo | Level, Spot, Date, Time, Car Plate, Description, Photo | Report submitted and confirmation shown | 10/10 Pass |
| TC02 – Missing Required Fields | Try submitting report with some fields left blank | Missing car plate or photo | Show error: 'Please fill in all fields and upload a photo' | 10/10 Pass |
| TC03 – No Network Connection | Submit report without internet | All valid fields but no connection | Show error: 'Network error. Please try again later.' | 10/10 Pass |
| TC04 – Temporary Parking Prompt | Submit a report and choose to request parking | Click 'Yes' on parking prompt | Assign temporary parking space | 10/10 Pass |
| TC05 – Cancel Temporary Parking | Submit report and decline parking | Click 'No' on parking prompt | Return to main screen without assigning parking | 10/10 Pass |

Table 6.2.6 Report Illegal Parking Test Case

| Register Visitor Test Case | | | | |
|---|---|---------------------------------|--|------------|
| Use Case: Register Visitor and Reference code Generator | | | | |
| Functional ID: F007 | | | | |
| Date Created: 24/4/2025 | | | | |
| Role: Resident | | | | |
| Test Case | Test Description | Input | Expected Output | Result |
| TC01 – Complete Registration | Submit valid visitor details | Full Name, IC, Car Plate, Phone | Show success and generate reference code | 10/10 Pass |
| TC02 – Missing Fields | Submit with one or more fields left empty | Missing IC or Phone | Show error: 'Please complete all fields' | 10/10 Pass |
| TC03 – Invalid Phone Number | Enter incorrectly | Letters or too few digits | Show error: 'Invalid phone' | 10/10 Pass |

| | | | | |
|---|---|--------------------------|---------------------------------------|------------|
| | formatted phone number | | number format' | |
| TC04 – Generate Reference Code | Register visitor and proceed | Valid visitor details | Unique reference code displayed | 10/10 Pass |
| TC05 – Copy Reference Code | Tap copy button on reference code screen | Copy action | Code copied to clipboard | 10/10 Pass |
| TC06 – Skip Reference Code | Tap 'Done' without copying | Tap Done | Return to main screen | 10/10 Pass |

Table 6.2.7 Register Visitor Test Case

| View My Reports Test Case | | | | |
|--|--|--|---|------------|
| Use Case: View My Reports | | | | |
| Functional ID: F008 | | | | |
| Date Created: 24/4/2025 | | | | |
| Role: Resident | | | | |
| Test Case | Test Description | Input | Expected Output | Result |
| TC01 – View Own Reports | Logged-in resident views report list | Resident email logged in | Show list of reports submitted by this user only | 10/10 Pass |
| TC02 – Report Status Display | Check status of each report | Submitted reports with different statuses | Statuses show as Pending, In Progress, or Resolved | 10/10 Pass |
| TC03 – No Reports Submitted | Resident with no past reports logs in | New resident account | Show message: 'No reports found' | 10/10 Pass |
| TC04 – View Report Details | Each item shows correct details and image | Click on report card | Display car plate, time, description, image, and status | 10/10 Pass |
| TC05 – Unauthorized Access Check | Ensure reports from other users are not visible | Attempt direct DB fetch (simulated) | Only user's own reports are shown | 10/10 Pass |

Table 6.2.8 View My Reports Test Case

| Manage Visitor Parking Test Case | | | | |
|----------------------------------|--|--------------------------------|--|------------|
| Use Case: Manage Visitor Parking | | | | |
| Functional ID: F009 | | | | |
| Date Created: 24/4/2025 | | | | |
| Role: Management | | | | |
| Test Case | Test Description | Input | Expected Output | Result |
| TC01 – Manual Spot Update | Update availability using floor plan selection | Click parking spot on map | Spot status toggled and saved in database | 10/10 Pass |
| TC02 – CSV Upload | Bulk update spot availability using CSV file | Upload valid CSV file | All data updated and reflected in UI | 10/10 Pass |
| TC03 – Invalid CSV Format | Upload CSV with incorrect headers or format | Malformed CSV | Show error: 'Invalid file format' | 10/10 Pass |
| TC04 – Real-Time Display | Updated spot shows correct status on screen | Manual or CSV update | Spot colour updated (green, red, blue) in floor plan | 10/10 Pass |
| TC05 – Retain Selections | Selections remain after view change | Switch between levels | Selected updates remain visible | 10/10 Pass |
| TC06 – No Spots Selected | Attempt update without selecting any spot | Click update with no selection | Show warning: 'No parking spots selected' | 10/10 Pass |

Table 6.2.9 Manage Visitor Parking Test Case

| Illegal Parking Report Test Case | | | | |
|---|--|----------------------|---|------------|
| Use Case: Management Illegal Parking Report | | | | |
| Functional ID: F010 | | | | |
| Date Created: 24/4/2025 | | | | |
| Role: Management | | | | |
| Test Case | Test Description | Input | Expected Output | Result |
| TC01 – View All Reports | Management accesses the report dashboard | Login as management | All submitted reports displayed with full details | 10/10 Pass |
| TC02 – Real-Time Report Fetch | Receive new reports | New report submitted | New report appears | 10/10 Pass |

| | | | | |
|-------------------------------------|--|------------------------------|--------------------------------|------------|
| | instantly from Firebase | | immediately in the list | |
| TC03 – Update to In Progress | Change status of report to In Progress | Click In Progress and Update | Status updated and saved | 10/10 Pass |
| TC04 – Update to Resolved | Change status of report to Resolved | Click Resolved and Update | Status updated and saved | 10/10 Pass |
| TC05 – Status Retains After Refresh | Reload screen after status update | Update and refresh | Updated status remains correct | 10/10 Pass |

Table 6.2.10 Illegal Parking Report Test Case

| NFC Car Details Test Case | | | | |
|---------------------------------|----------------------------------|----------------------------------|---|------------|
| Use Case: NFC Car Details | | | | |
| Functional ID: F011 | | | | |
| Date Created: 24/4/2025 | | | | |
| Role: Management | | | | |
| Test Case | Test Description | Input | Expected Output | Result |
| TC01 – Complete Submission | Submit all required car details | Plate, Model, Color, Parking Lot | Proceed to NFC writing screen | 10/10 Pass |
| TC02 – Missing Fields | Submit form with incomplete data | Leave one or more fields empty | Show error: 'Please fill in all fields' | 10/10 Pass |
| TC03 – Invalid Car Plate Format | Input non-standard plate format | Special characters or too short | Show error: 'Invalid plate number format' | 10/10 Pass |
| TC04 – Reset After Success | Return to form after writing NFC | Completed write and back | Fields reset and ready for new input | 10/10 Pass |

Table 6.2.11 NFC Car Details Test Case

| NFC Tag Writer Test Case | | | | |
|--------------------------|---------------------------------------|---------------------------|------------------------------------|------------|
| Use Case: NFC Tag Writer | | | | |
| Functional ID: F012 | | | | |
| Date Created: 24/4/2025 | | | | |
| Role: Management | | | | |
| Test Case | Test Description | Input | Expected Output | Result |
| TC01 – Successful Write | Tag placed correctly and data written | Valid NFC tag near device | UI turns green, data stored in tag | 10/10 Pass |

| | | | | |
|---------------------------|------------------------------------|----------------------------|--|------------|
| TC02 – Tag Not Detected | NFC is on but tag not close enough | No tag detected | Show prompt: 'Please tap your NFC tag again' | 10/10 Pass |
| TC03 – NFC Disabled | Attempt to write with NFC off | NFC off | Show error: 'Please enable NFC in settings' | 10/10 Pass |
| TC04 – Missing Permission | Application lacks NFC permission | Permissions not granted | Show error: 'NFC permission required' | 10/10 Pass |
| TC05 – Cancel Mid Write | User exits before write completes | Exit activity during write | Return to previous screen without writing | 10/10 Pass |

Table 6.2.12 NFC Tag Writer Test Case

| NFC Tag Reader Test Case | | | | |
|---------------------------|---|------------------------------|--|------------|
| Use Case: NFC Tag Reader | | | | |
| Functional ID: F013 | | | | |
| Date Created: 24/4/2025 | | | | |
| Role: Management | | | | |
| Test Case | Test Description | Input | Expected Output | Result |
| TC01 – Successful Read | Place a valid NFC tag near the device | Valid written tag | Display all stored car details | 10/10 Pass |
| TC02 – Tag Not Detected | Place tag too far from reader | Weak tap or distance too far | Show message: 'Tag not detected' | 10/10 Pass |
| TC03 – Unwritten Tag | Scan an empty tag | Blank NFC tag | Show error: 'No data found on tag' | 10/10 Pass |
| TC04 – NFC Disabled | Scan while NFC is off | NFC turned off | Show prompt: 'Please enable NFC to scan tag' | 10/10 Pass |
| TC05 – Missing Permission | Try to read without NFC permissions granted | Permissions off | Show error: 'NFC permission required' | 10/10 Pass |

Table 6.2.13 NFC Tag Reader Test Case

6.3 Project Challenge

The project encountered several challenges in implementing the auto-assign carpark feature, particularly in the development of the view map functionality. While the system successfully assigns parking spots and displays them on a visual floor plan, it does not support GPS-based navigation. This is mainly due to the limitations of GPS in indoor environments such as covered or basement parking areas, where signals are often weak or completely blocked.

One of the key challenges involved the handling of multi-level parking layouts. Each level requires its own dedicated floor plan and coordinate mapping, which increases the system's complexity. Furthermore, GPS is not capable of determining vertical positioning, meaning it cannot detect which level a user is currently on. For instance, if a user is on Level 2, the GPS cannot accurately differentiate it from Level 1 or Level 3. This makes GPS unsuitable for navigating within multi-level parking structures.

As a result, the system relies entirely on a pre-designed static map, where each parking spot is manually positioned using hardcoded X and Y coordinates. While this approach allows visual representation of assigned spots, it lacks flexibility and scalability and may not always provide clear guidance to users who are unfamiliar with the parking layout.

In summary, the inability to use GPS indoors, the limitations in detecting vertical levels, the reliance on fixed coordinates, and the complexity of managing multiple floor plans posed significant technical and design challenges during the development of this feature.

6.4 Objective Evaluation

This section evaluates the extent to which the project objectives have been achieved based on the implemented features and functionalities in the final system.

Objective 1: To implement secure NFC authentication in a parking management system mobile application using Android Studio

This objective has been successfully achieved. The system integrates NFC technology through Android Studio, allowing authorized vehicles to be identified using NFC-enabled parking stickers. The mobile application reads and verifies the data from NFC chips attached to each sticker, ensuring that only authorized vehicles can be recognized by the system. This helps prevent unauthorized access and illegal duplication. The implementation has been tested and validated during real-time scanning and verification within the application.

Objective 2: To develop an interface that simplifies information entry and access to dedicated floor plans

This objective has also been achieved. The mobile application provides a user-friendly interface that replaces traditional manual processes with an automated system. When a resident or visitor needs a parking space, the system automatically assigns an available slot and presents a pre-designed floor plan for visual navigation. This eliminates the need for physical forms or verbal directions and reduces the likelihood of user error. The feature has been integrated and is fully functional within the application.

Objective 3: To create a dashboard in the mobile application for monitoring occupancy and verifying parking stickers via NFC

This objective is fulfilled through the implementation of a management dashboard within the mobile application. The dashboard allows administrators to view real-time occupancy data, check the availability of parking spots, and verify NFC stickers for security purposes. Additionally, the system supports illegal parking reports submitted

by residents, which can be reviewed and managed by the parking authority through the same dashboard. These features have been implemented and tested to ensure they support efficient parking management and enforcement.

All the defined objectives of this project have been successfully achieved. The features are fully implemented and verified through testing and actual application interaction, demonstrating the system's ability to improve parking security, simplify navigation, and enhance administrative oversight.

6.5 Concluding Remark

In conclusion, the development and evaluation of the Parking Management System with Mobile NFC Authentication demonstrate the successful realization of the project's goals. The system effectively integrates key technologies such as Firebase for real-time database management and authentication, and NFC for secure vehicle identification. Through a comprehensive series of functional tests, performance evaluations, and module validations, the system has proven to be both functional and reliable.

fo

The implemented modules—including visitor registration, parking space assignment, illegal parking reporting, and NFC tag operations—work cohesively to provide a seamless user experience for residents, visitors, and management alike. Testing has shown that image loading, network responsiveness, and database performance all fall within acceptable operational thresholds, ensuring smooth real-time interactions.

Despite technical challenges such as limitations in GPS usage for indoor navigation and the complexity of handling multi-level floor plans, the system has successfully overcome these issues through design workarounds, such as the use of static floor plans with visual cues.

Overall, the Parking Management System not only meets its initial objectives but also lays a strong foundation for future enhancements. Its robust functionality, practical performance, and secure authentication mechanisms make it a viable and scalable solution for real-world residential parking environments.

CHAPTER 7

Conclusion

7.1 Conclusion

The Parking Management System with Mobile NFC Authentication was developed to address the real-world challenges of managing residential parking efficiently and securely. Through a thoughtful integration of Android Studio, Firebase, and NFC technology, the system successfully delivers tailored functionalities for three key user groups: residents, visitors, and management.

Throughout this report, each module and feature—from secure login and email verification to visitor registration, illegal parking reporting, and NFC-based tag verification has been clearly designed to improve usability, enforce parking rules, and reduce manual errors. The system empowers residents to actively manage their parking and visitor needs, allows visitors to gain seamless entry with a reference code, and gives management the tools to verify parking through NFC, review reports, and control parking lot availability in real time.

Functionality has been supported by a reliable backend architecture using Firebase Firestore and Storage, which ensures real-time updates, fast data retrieval, and secure handling of sensitive information. The system's performance has been validated through comprehensive testing, demonstrating fast image loading, responsive auto-parking assignment, and acceptable database processing times, even with bulk data.

Furthermore, activity diagrams, use case models, and database structures provided in the report illustrate the system's logical flow and integrity, ensuring scalability and adaptability for future upgrades.

In conclusion, the project has met all its stated objectives by delivering a practical and robust solution for residential parking management. It enhances

convenience, enforces accountability, and leverages smart technology to transform traditional parking systems into a smarter, more secure experience for all users involved.

7.2 Recommendation/ Future Work

One of the key areas identified for future improvement is the parking floor map feature. While the current implementation effectively shows visitors their assigned parking lot and general direction, there are several opportunities to enhance its usability and visual guidance.

Firstly, the map could be made interactive by allowing users to zoom, pan, or tap on parking spots to view more details. Currently, the static layout provides basic direction but does not offer a dynamic experience. Adding animation or a mini-step indicator could help guide users more clearly across multiple levels.

Another major enhancement would be the integration of indoor positioning technology, such as Bluetooth beacons or Wi-Fi-based location tracking, which could simulate real-time navigation inside the parking complex. Although GPS is not suitable for indoor environments, these alternatives could help visitors locate their parking spots more intuitively.

Lastly, the map design can be made more visually appealing with 3D layout elements, custom icons, and multi-language support to accommodate a diverse group of users.

By enhancing the floor map in these ways, the parking system can become even more intuitive, user-friendly, and accessible for all users, especially for visitors unfamiliar with the layout.

REFERENCES

- [1] NFC Forum. "NFC Technology." *NFC Forum*, <https://nfc-forum.org/learn/nfc-technology/>. Accessed May 7, 2025.
- [2] S. Online, "Too many cars, not enough space," *The Star*, Nov. 13, 2023. [Online]. Available: <https://www.thestar.com.my/news/nation/2023/11/14/too-many-cars-not-enough-space>
- [3] "Apartment Residents Plagued by Parking Shortage," *The Sun*, Nov. 2023. [Online]. Available: <https://thesun.my/local-news/apartment-residents-plagued-by-parking-shortage-GL11830767>.
- [4] "Parking Issues: Pet Problems Top Tiffs Among Neighbours," *The Star*, Nov. 27, 2023. [Online]. Available: <https://www.thestar.com.my/news/nation/2023/11/27/parking-issues-pet-problems-top-tiffs-among-neighbours>.
- [5] ZKTeco Malaysia, "Barrier Gate and RFID Solutions," [Online]. Available: <https://www.zkteco.com.my/landing/barrier-gate-rfid-lpr>.
- [6] TimeTec, "Smart Parking Solutions," [Online]. Available: <https://www.timeteccloud.com/landing/parking/0523>.
- [7] "Veemios," *Veemios.com*, 2022. <https://veemios.com/> (accessed Sep. 04, 2024).
- [8] D. Sethia, D. Gupta, and H. Saran, "NFC Secure Element-Based Mutual Authentication and Attestation for IoT Access," *IEEE Transactions on Consumer*

REFERENCES

Electronics, vol. 64, no. 4, pp. 470–479, Nov. 2018, doi: <https://doi.org/10.1109/tce.2018.2873181>.

[9] W. Li, W. Wong, and K. Chow, "NFC-Based E-Ticketing System with OTA Ticket Provisioning and Offline Authentication," *IEEE Transactions on Consumer Electronics*, vol. 60, no. 4, pp. 411–418, Nov. 2014, doi: 10.1109/TCE.2014.6782911. Available: <https://ieeexplore.ieee.org/document/6782911>.

[10] The4, "Pros and Cons of NFC Tags - Infographic," *NFC Tagify*, Aug. 26, 2023. <https://nfctagify.com/blogs/news/pros-and-cons-of-nfc-tags-infographic>

[11] QR Code Matic, "Advantages and Disadvantages of QR Codes," *QR Code Matic*, 2024. <https://www.qrcodematic.com/advantages-and-disadvantages-of-qr-codes>.

[12] J. L. Perez, "Biometric Authentication: Advantages and Disadvantages," *Recordia*, Oct. 07, 2022. <https://recordia.net/en/understanding-biometric-authentication-advantages-and-disadvantages/>

[13] "What is Password-Based Authentication?," *www.descope.com*, Apr. 17, 2023. <https://www.descope.com/learn/post/password-authentication#the-disadvantages-of-password-authentication>

[14] "iNeighbour," *iNeighbour*, 2024. <https://www.i-neighbour.com/landing/0424-1.php#Visitor> (accessed Sep. 04, 2024).

[15] "Visitor Management System | iNeighbour," *I-neighbour.com*, 2024. <https://www.i-neighbour.com/visitor-management-system/>

REFERENCES

- [16] “Smart Parking (LPR) - Sunway Pyramid,” *www.sunwaypyramid.com*.
<https://www.sunwaypyramid.com/smartparking>
- [17] R. Raghavan, “The Importance of User-Centered Design in Mobile App Development,” *Medium*, Aug. 25, 2023. <https://medium.com/@rohithraghavan99/the-importance-of-user-centered-design-in-mobile-app-development-af37f10544dd>
- [18] GeeksforGeeks, “Waterfall Model Software Engineering,” *GeeksforGeeks*, Apr. 26, 2025. <https://www.geeksforgeeks.org/waterfall-model/>
- [19] GeeksforGeeks, “Agile development models Software engineering,” *GeeksforGeeks*, Apr. 11, 2025. <https://www.geeksforgeeks.org/software-engineering-agile-development-models/#what-is-agile-model>
- [20] “Prototype Model - javatpoint,” *www.javatpoint.com*.
<https://www.javatpoint.com/software-engineering-prototype-model>

POSTER



Faculty of Information Communication and Technology
Bachelor of Information Technology (Hons) Information System Engineering

Parking Management System with Mobile NFC Authentication



PROJECT BACKGROUND

Residential parking areas often face challenges such as unauthorized parking, inefficient space allocation, and limited management oversight. These issues cause frustration for both residents and management, as well as raise security concerns. Existing solutions often lack comprehensive features that address these issues effectively, especially in integrating management, resident, and visitor functionalities into one system.

OBJECTIVE

1. Implement NFC authentication to enhance security and prevent unauthorized parking access.
2. Develop a user-friendly interface for easy information entry and parking navigation.
3. Create a management dashboard to monitor occupancy, verify stickers, and manage parking issues.

CONCLUSION

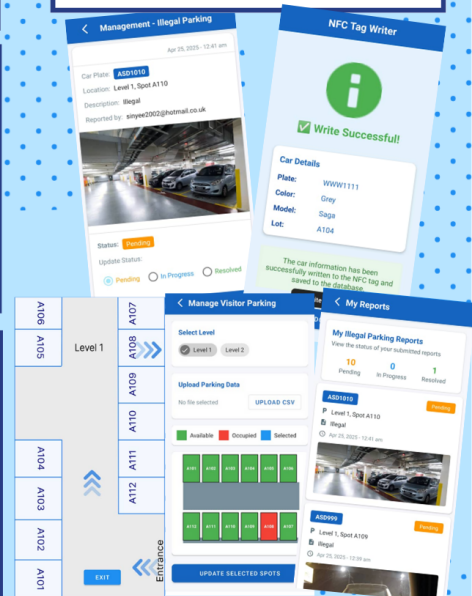
This mobile application offers a technology-driven solution to common residential parking issues. By integrating NFC technology, real-time monitoring, and comprehensive communication features, it improves parking security, enhances user experience, and optimizes space utilization. The app will be measured by its ability to reduce unauthorized parking incidents and improve overall parking efficiency in residential areas.

OBJECTIVE EVALUATION

All project objectives were successfully achieved. The system implements secure NFC authentication, a user-friendly floor map for navigation, and a real-time admin dashboard for monitoring and management. Testing confirmed that all features function reliably, improving parking efficiency, security, and user experience.

FUNCTIONALITY

- Mobile NFC Authentication for Sticker
- Resident Reporting Illegal Parking
- Visitor Parking Reservation
- Auto-Assigned Parking
- Management Monitoring



PROJECT DEVELOPER:
NG SIN YEE



PROJECT SUPERVISOR:
MS TAN LYK YIN

