JAIN DEEMED-TO-BE UNIVERSITY | FACULTY OF ENGINEERING AND TECHNOLOGY

Department of Computer Science and Engineering

Global Campus, Jakkasandra Post, Kanakapura Taluk, Ramanagara District, Pin Code: 562 112

2022-2023

A Project Report on

"SPYPARK – THE INNOVATIVE MOBILE APPLICATION FOR ONLINE PARKING RESERVATION"

Submitted in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

SOFTWARE ENGINEERING

Submitted by

ANUSHA SARKAR
19BTRSE002

RANI KUMARI
19BTRSE025

UDDESHYA GORAI
19BTRSE034

Under the guidance of

Prof. Karthikeyan S.

Assistant Professor
Department of Computer Science & Engineering
Faculty of Engineering & Technology
JAIN DEEMED-TO- BE UNIVERSITY

JGi JAIN DEEMED-TO-BE UNIVERSITY | FACULTY OF ENGINEERING AND TECHNOLOGY

Department of Computer Science and Engineering

Global Campus, Jakkasandra Post, Kanakapura Taluk, Ramanagara District, Pin Code: 562 112

CERTIFICATE

This is to certify that the project work titled **"SPYPARK – THE INNOVATIVE MOBILE APPLICATION FOR ONLINE PARKING RESERVATION"** is carried out by **ANUSHA SARKAR (19BTRSE002), RANI KUMARI (19BTRSE025), UDDESHYA GORAI (19BTRSE034),** a bonafide students of Bachelor of Technology at the Faculty of Engineering & Technology, Jain (Deemed-to-be) University, Bangalore in partial fulfillment for the award of degree in Bachelor of Technology in **Software Engineering**, during the year **2022-2023**

Mr.S. Karthikeyan Guide Assistant Professor, Dept. of CSE-SE, Faculty of Engineering & Technology, Jain (Deemed-to-be) University Date:	**Dr. Chandramma R** Program Head, Dept. of CSE-SE Faculty of Engineering & Technology, Jain (Deemed-to-be) University Date:	**Dr. Geetha G** Director, School of Computer Science & Engineering, Jain (Deemed-to-be) University Date:

Name of the Examiner	Signature of Examiner
1.	
2.	

DECLARATION

We, **ANUSHA SARKAR (19BTRSE002), RANI KUMARI (19BTRSE025), UDDESHYA GORAI (19BTRSE034),** are students of eighth semester B.Tech in **Software Engineering**, at Faculty of Engineering & Technology, **Jain** (**Deemed-to-be) University**, hereby declare that the project titled **"SPYPARK – THE INNOVATIVE MOBILE APPLICATION FOR ONLINE PARKING RESERVATION"** has been carried out by us and submitted in partial fulfilment for the award of degree in **Bachelor of Technology in Software Engineering** during the academic year **2022-2023**. Further, the matter presented in the project has not been submitted previously by anybody for the award of any degree or any diploma to any other University, to the best of our knowledge and faith.

Name: ANUSHA SARKAR

Signature

USN: 19BTRSE002

Name: RANI KUMARI

Signature

USN2: 19BTRSE025

Name: UDDESHYA GORAI

Signature

USN: 19BTRSE034

Place: Bangalore

Date: 26/05/2023

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of a large number of individuals who have been responsible for the successful completion of this project work.

First, we take this opportunity to express our sincere gratitude to ***Faculty of Engineering and Technology, Jain (Deemed-to-be) University,*** *for providing us with a great opportunity to pursue our Bachelor's Degree in this institution.*

In particular we would like to thank ***Dr. Geetha. G, Director, School of Computer Science and Engineering, and Dr. R. Chandramma Program Head, Department of Software Engineering, Jain (Deemed-to-be University),*** *for their constant encouragement and expert advice.*

We would like to thank our guide ***Mr. Karthikeyan S., Assistant Professor, Department of Computer Science and Engineering, Jain (Deemed-to-be University),*** *for sparing his valuable time to extend help in every step of our project work, which paved the way for smooth progress and fruitful culmination of the project.*

We would like to thank our Project Coordinator ***Dr. R. Chandramma*** *and all the staff members of Computer Science & Engineering (SE) for their support.*

We are also grateful to our family and friends who provided us with every requirement throughout the course.

We would like to thank one and all who directly or indirectly helped us in completing the Project work successfully.

Signature of Students

Anusha Sarkar (19BTRSE002)

Rani Kumari (19BTRSE025)

Uddeshya Gorai (19BTRSE034)

TABLE OF CONTENTS

CERTIFICATE	**ii**
DECLARATION	**iii**
ACKNOWLEDGEMENT	**iv**
TABLE OF CONTENT	**v**
ABSTRACT	**vii**
LIST OF FIGURES	**viii**
NOMENCLATURES USED	**ix**

CHAPTER 1

INTRODUCTION

1.1 Overview	10
1.2 Problem Definition	12
1.3 Objectives	12
1.4 Methodology	13
1.5 Hardware and Software Tools Used	14

CHAPTER 2

LITERATURE SURVEY

2.1 Related Work	15
2.2 Existing Work	17
2.3 Limitations of Existing Work	17
2.4 Proposed Methodology	18

CHAPTER 3

SYSTEM DESIGN

3.1 Architecture Diagram	20
3.2 Use Case Diagram	22
3.3 Sequence Diagram	23
3.4 Activity Diagram	24

3.5. Class Diagram 25

CHAPTER 4

TOOLS DESCRIPTION

4.1 Hardware Requirements 27

4.2 Software Requirements 27

Chapter 5

IMPLEMENTATION

5.1 Description 28

5.2 Codebase Structure Breakdown 31

5.3 Code Snippets 32

Chapter 6

SYSTEM TESTING AND RESULTS 59

Chapter 7

CONCLUSION AND FURTHER SCOPE 65

REFERENCES 66

APPENDIX - I 68

APPENDIX - II 69

ABSTRACT

Parking has become difficult, due to the rising number of vehicles, especially in congested areas such as public transportation and galleria. Most parking lots still use traditional parking systems that involve drivers obtaining a ticket upon entering the lot, searching for a vacant parking space, paying at the counter, and scanning the ticket QR code at the time of exit. This process is time-consuming and frustrating for drivers, especially during peak periods. To address this issue, an online parking reservation system is needed to manage parking spaces. This system allows drivers to reserve parking spaces in advance from any location, providing greater convenience and reducing the time spent looking for parking. The online reservation system enables users to reserve parking spots in nearby lots and other services, making it easier for them to find a parking spot. This mobile application provides notifications in advance of available parking spaces and reserves a specific parking place, reducing parking lot congestion and saving time and money for drivers. It also helps to promote a clean and green environment. The system allows administrators to manage parked vehicles by adding new vehicles and removing old ones, providing greater control over the parking lot. Although most financial transactions are currently done with cash or online, a digital parking system would be more beneficial than the current manual parking system.

Keywords — Manual parking system, online reservation system, parking spot, convenience, time-saving, traffic congestion, clean environment, administrator control.

LIST OF FIGURES

Fig. No.	Description of Figure	Page No.
3.1.1	Architecture Diagram of Server	20
3.1.2	Architecture Diagram	21
3.2.1	User Use Case Diagram	22
3.2.2	Admin Use Case Diagram	22
3.3	Sequence Diagram	23
3.4.1	User Activity Diagram	24
3.4.2	Admin Activity Diagram	25
3.5	Class Diagram	26
5.2	A screenshot of the codebase breakdown of the Spypark Admin and User app	31
6.1.1	A screenshot of Splash Screen	60
6.1.2	A screenshot of Login Page	60
6.1.3	A screenshot of Sign Up Page	61
6.1.4	A screenshot of Forgot Password Page	61
6.1.5	A screenshot of Welcome Page	62
6.1.6	A screenshot of Parking Areas	62
6.1.7	A screenshot of Parking Area Vacancy	63
6.1.8	A screenshot of Parking Area Vacancy	63
6.1.9	A screenshot of Items Page	64
6.1.10	A screenshot of all Bookings	64

NOMENCLATURE USED

GUI	Graphical User Interface
SDLC	Software Development Life Cycle
ASD	Agile Software Development
ANDROID	● Android - SDK ● Java ● XML

CHAPTER – 1

INTRODUCTION

Today, the increased number of vehicles creates challenges for parking systems. Parking in crowded areas, such as public transportation and galleria, can be difficult for drivers due to limited available space. Most parking lots still function using the traditional parking system, which requires drivers to obtain a ticket at the entrance, search for a vacant space, pay for the parking, and scan the paid ticket QR code at the exit. Drivers often must spend more time and effort finding a vacant parking spot, especially during festivals, vacations, and weekends.

During these busy periods, it can be challenging for drivers to find a safe and available parking spot for their vehicles. To address this problem, online parking reservation systems have emerged as a promising solution. These systems allow drivers to reserve a parking spot in advance from any location, providing them with increased convenience and reducing the time spent searching for parking. Drivers can choose a specific place in parking lots based on the size of their vehicle. Once reserved, it will be unavailable to others.

With the online reservation system, parking spots in the nearby lot, as well as other services, can be reserved, making it simpler for users. There is no established method for locating parking spaces in modern parking lots. For many people, searching for a parking spot in a large city is a daily inconvenience that consumes a significant amount of time. The benefit of this mobile application is that it reserves a specific parking place and uses a mobile application to notify consumers in advance of available parking spaces. This application will provide the users with a platform where they will be able to decide where and which parking slot to book for parking their vehicles.

1.1 Overview

An online parking reservation mobile application is a digital platform that allows users to search, book, and pay for parking spaces using their mobile devices. These applications aim to simplify and streamline the process of finding parking by providing users with real-time information, convenience, and secure transactions. Here is an overview of the key features and benefits of online parking reservation mobile applications:

1. **Parking Spot Search:** Users can search for available parking spots based on their desired location, date, and time. The application provides real-time information on parking availability, including details such as pricing, distance from the user's current location, and specific amenities (e.g., electric vehicle charging stations, disability-friendly spaces).
2. **Booking and Reservation:** Users can select their preferred parking spot and make a reservation directly through the mobile application. This feature ensures that a designated parking space is reserved for the user, eliminating the hassle of searching for parking upon arrival.
3. **Digital Payments:** Online parking reservation apps facilitate cashless transactions, allowing users to make payments securely within the application. This feature often supports various payment methods such as credit/debit cards, mobile wallets, or pre-paid parking credits.
4. **Cancellation and Modifications:** If users need to modify or cancel their parking reservation, online parking reservation apps often provide flexibility to make changes through the application. This allows for convenient management of parking plans.
5. **User Reviews and Ratings:** Users can provide feedback and ratings for parking facilities they have used, helping others make informed decisions when selecting parking spots. This feature contributes to a community-driven approach to sharing experiences and enhancing the overall quality of parking services.
6. **Integration with Parking Facilities:** Online parking reservation apps collaborate with parking facilities, both public and private, to offer a wide range of parking options. This includes parking garages, lots, and even private driveways, expanding the choices available to users.

Overall, online parking reservation mobile applications aim to save time, reduce stress, and enhance the parking experience by providing users with easy access to available parking spaces, convenient booking and payment options, and relevant information to ensure a smooth parking process.

1.2 Problem Definition

The parking place is very important all over the world, especially in the cities of the countries. Every day thousands of car drivers spend a lot of time to find where to park. The result of this situation is theft in urban areas, increasing traffic congestion, and frustration among drivers. To solve this problem, the implementation of an Online Parking Reservation System for managing parking places is mandatory. It will allow the drivers to Reserve a parking place on the Platform anytime, anywhere.

The online reservation system to reserve parking spaces in the immediate parking and additional services. It is designed to make it easier for people to book parking spaces online. There is no standard system to check for parking spaces in today's parking lots. Searching for a vacant parking space in a metropolitan area is a daily concern for most people and it is time-consuming. It commonly results from more traffic load and air pollution in certain areas only for an available parking space.

Admin can manage the parked vehicle by adding incoming vehicles and removing outgoing vehicles. Although financial transactions are done by cash in hand or online transaction, we believe this project would help a lot in parking using computerized methods instead of the old parking system.

1.3 Objective(S)

By paying, we have the option to park our vehicle in our designated slot, which eliminates any towing issues. This ensures that our vehicle is securely parked, reducing the risk for the vehicle owner. In the event of any damages or issues with the vehicle, the responsibility falls on the parking management to address and resolve.

Given the current global security threats, including increasing cases of robberies and vehicle-related incidents like bomb blasts, implementing a robust system can help in tracking and recording such incidents.

Developing efficient and user-friendly software is crucial, as it allows for quick record maintenance and facilitates easy tracking of individuals. Additionally, it enables us to determine the availability of parking spaces promptly and enhances the overall experience for visitors.

- Providing a simple Android application for parking vehicles.
- Booking for parking slot at home or anywhere.
- Car search nearby places using Google Maps.
- Easy payment system.
- Parking owners can add their parking places.
- Make it easy to automate parking owners and customers.

1.4 Methodology

Methodology refers to the systematic approach or set of methods used to conduct research, gather data, and analyze information to achieve specific objectives or answer research questions.

As this is an android project, the whole codebase will be done in the android studio in which java language will be used in the backend and for the frontend xml, certain open-source dependencies, libraries will be used. For the database integration, Firebase will be used for whole storage and retrieval of data. Images will be stored in Firebase Fire store and other data such as user details, booking and end time and parking places will be stored in Firebase Realtime Database. Various dependencies will be used from open-source directories present. After the completion of the project, the users will be able to view their booking details and other previous records.

In the context of developing an online parking reservation mobile application, the methodology typically involves several key steps. This may include requirements gathering, market research, conceptualization, design, development, testing, deployment, and ongoing evaluation. The methodology ensures a structured and organized approach to the application development process, enabling efficient collaboration among team members, adherence to project timelines, and the delivery of a high-quality and user-friendly mobile application that meets the needs and expectations of the target users.

1.5 Hardware and Software Used

Since the project based on use of Android Studio, the hardware requirements are bare minimum.

Have a look at the Hardware requirements for the project:

- The Hardware is required to test and run things locally before committing changes to the main team branch.
- A bare minimum configuration of 4 GB RAM with 10+ GB of free disk space would be more than sufficient.
- Operating Systems are user preference. Whichever OS the user is comfortable with can be used.
- A good internet connection is required.
- GitHub Software and local Git repository should be present in the machines.
- Knowledge on application development.
- Knowledge on developing applications.
- Text/Code Editor: Android Studio
- Language: Java, Xml
- Database: Firebase

CHAPTER – 2

LITERATURE SURVEY

Online parking reservations are becoming increasingly popular to save time and reduce stress for drivers. A literature survey on this topic would likely cover research related to the benefits of online parking reservations, the factors that influence users' adoption of such services, and the design and implementation of online parking reservation systems.

'Rosalyn R. Porle and Nursyafiqah Nabilah Mohd Saiful' [1] suggested, by enabling drivers to book parking slots in advance, the Android application for parking helps them save time and reduces congestion in parking lots. Although there have been various studies on smart parking systems, minimum research is focused on mobile applications like this one. According to the survey, 85.7 percent of the participants found the app easy to use, while 14.3 percent found it average in terms of usability. During holidays, weekends, and festivals, these issues become more prevalent, making it difficult for drivers to find available and safe parking spots.

'Lai Tu et al' [2] proposed, that many contemporary parking lots now incorporate various IoT technologies, including license plate recognition and automatic entrance/exit control, to enhance the ease of drivers and parking lot supervisors, as well as to analyze parking flow patterns and predict occupancy based on historical data.

'Hongwei Wang and Wenbo He' [3] presented their idea, that finding a parking spot in metropolitan areas, particularly during peak traffic times, can be a challenging task for drivers. A Reservation-based Smart Parking System (RSPS) prototype has been developed, which enables drivers to efficiently locate and book unoccupied parking spaces. This cyber-physical system can be accessed by drivers through their communication devices. It helps to minimize traffic congestion and air contamination caused by drivers persistently searching for an available parking space in a particular area.

'Raji C.G. et al' [4] proposed, despite significant research on the development of smart parking systems, several of these systems do not effectively tackle the problem of real-time identification of available parking spaces and automated parking fee collection. With the population and economy expanding, the number of vehicles is also increasing. In metropolitan regions, it is estimated that over 40% of total traffic density is due to vehicles cruising for parking. This continuous circling of a specific area contributes to traffic congestion and air pollution.

'P. Sheelrani et al' [6] proposed, that the virtual AGV is a fully automatic parking system that is a result of theoretical research conducted to address the parking management issues prevalent in traditional parking lots, such as insufficient parking spaces, subpar service quality, and low user satisfaction. This system introduces a novel dynamic priority approach that efficiently resolves the deadlock conflicts responsible for the backlog of parking AGVs. Currently, mechanical parking garages and automatic guided transport vehicle (AGV) conveyance parking systems have been developed.

'Gayatri N. Hainalkar and Mousami S. Vanjale' [7] suggested that the rapid urban population growth has created a major parking problem in most major cities around the world. Finding proper parking space for our vehicles can be a significant source of frustration. To address this issue, a Slot Allotment based Intelligent Parking System has been implemented, allowing Android users to book parking slots in two modes: advanced and current booking.

'M. Karthi and Preethi Harris' [8] proposed, as cities move towards becoming smarter, a variety of smart applications are being developed. The IoT has been a significant technological and economic advancement in the information industry after the Internet. However, the inability to pre-book public parking spaces has led to a decline in the quality of urban mobility.

'Abrar Fahim et al' [9] proposed, that during peak hours or traffic congestion, drivers in densely populated cities frequently struggle to find available parking spaces. This is done using an Android app that enables drivers to initiate requests for parking slots. Currently, the common method of searching for parking spaces is by doing a manual search, in which drivers rely on luck and experience to find an available spot on the street.

'Arwa A. Moosa and Mays Afif Anaee' [10] presented a method, to tackle the issue of locating a safe and vacant parking spot in busy urban areas, a Smart Parking System (ISPS) was created. The ISPS incorporates several features such as a rotary parking system to alleviate congestion, car number plate recognition for identification, and a Fiber-Optic vibration fence for added security. The system was engineered to limit human intervention and relies on pre-programmed controllers for access control in restricted zones.

'PK Jogada and V Warad ' [11] proposed, a smart parking application that enables users to locate and secure empty parking spots via an Android Application or Embedded Hardware. offers two booking modes - advanced and Current. To ensure dynamic Slot Allotment monitoring, the server is utilized.

2.1 Existing work

There are several online parking reservation mobile applications available in the market that aim to simplify the process of finding and reserving parking spaces. Some well-known examples include:

- **ParkMobile:** ParkMobile is a popular parking app that allows users to find, reserve, and pay for parking spots in various cities. It provides real-time availability, navigation to parking locations, and digital payment options.
- **SpotHero:** SpotHero enables users to find and reserve parking spaces in advance, providing access to a network of parking lots, garages, and valet services. It offers features like price comparisons, discounted rates, and digital parking passes.
- **ParkWhiz:** ParkWhiz offers a mobile app for finding and booking parking spots in advance. It provides options for hourly, daily, and monthly parking, along with features like real-time availability, price comparisons, and mobile entry/exit.
- **JustPark:** JustPark allows users to find and reserve parking spaces in various locations, including private driveways, parking lots, and garages. The app provides information on availability, pricing, and navigation to the chosen parking spot.
- **BestParking:** BestParking helps users find parking facilities, compare prices, and reserve parking spaces in major cities. It provides information on rates, availability, and special deals, allowing users to save time and money.

2.2 Limitations of Existing Work

While online parking reservation mobile applications offer convenience and streamline the process of finding and reserving parking spaces, they may have certain limitations. Here are some common limitations associated with existing online parking reservation mobile applications:

1. **Limited Coverage:** Online parking reservation apps may have limited coverage, especially in smaller cities or less-populated areas. The availability of parking spaces and participating parking facilities can vary greatly depending on the location.
2. **Parking Facility Integration:** Some parking facilities may not be integrated into the online reservation system, making it necessary for users to manually check availability or make reservations through other means.
3. **Real-time Availability:** While many apps provide real-time information about parking availability, the accuracy and timeliness of this data can vary. There may be instances where the displayed availability doesn't match the actual availability on-site due to delays in updating the information.
4. **User Experience:** The user experience of different parking reservation apps can vary in terms of app design, usability, and performance. Some apps may have complex navigation, slow response times, or unintuitive interfaces, which can impact the overall user experience.
5. **Integration with Payment Systems:** Issues may arise with payment systems integration, causing inconvenience or failed transactions for users. Problems like payment gateway errors or limited payment options may hinder the smooth reservation and payment process.
6. **Limited Customization:** Some apps may lack customization options, restricting users from specifying specific requirements such as charging stations for electric vehicles, disability-friendly parking spaces, or other specialized needs.

2.3 Proposed Methodology

The proposed project aims to address the challenges of finding parking spaces in busy commercial areas by introducing an online parking reservation system. This system provides clients with the convenience of reserving a parking space through a mobile application. Users can view available parking spaces and book a space for a specific time slot, which will be marked as yellow and reserved for the user during the specified time. The system also allows users to cancel their bookings and make payments online using credit cards, UPI, and other online payment methods. After making a payment, users receive an email confirmation with a unique parking number. Overall, this smart parking booking system offers an easy and efficient solution for finding and reserving parking spaces in busy areas.

The system shows three components in this architecture including parking zones, users, and the admin. As a result, the state of parking resources is changed by user's parking decisions. The management system broadcasts live parking availability information to users (also drivers). Upon receiving parking information, the user selects desired parking lot and reserves a space. User can have their username, login id, phone number, email, and address. Admin can collect the whole data from the database system.

User-defined API software architecture designed mainly shows Android applications, user applications, and functions allocated to the central control system and the structure of the host system application point location.

The first is to create a user account to be able to use the service provided. After creating an account, the user username, and password can log in to his mobile phone.

The user can select the proper parking lot and check the availability of free spaces are available so the user can go to a space reservation. A user is only allowed to reserve a space.

Admin-side functionality:

- Add new parking areas and spaces
- Modify parking rates
- Generate reports on occupancy and revenue
- Manage user accounts and transactions
- Notification for system updates and issues

User-side functionality:

- View parking history
- View parking rates
- Notification for booking confirmation and parking expiration

CHAPTER – 3

SYSTEM DESIGN

3.1 Architecture Diagram

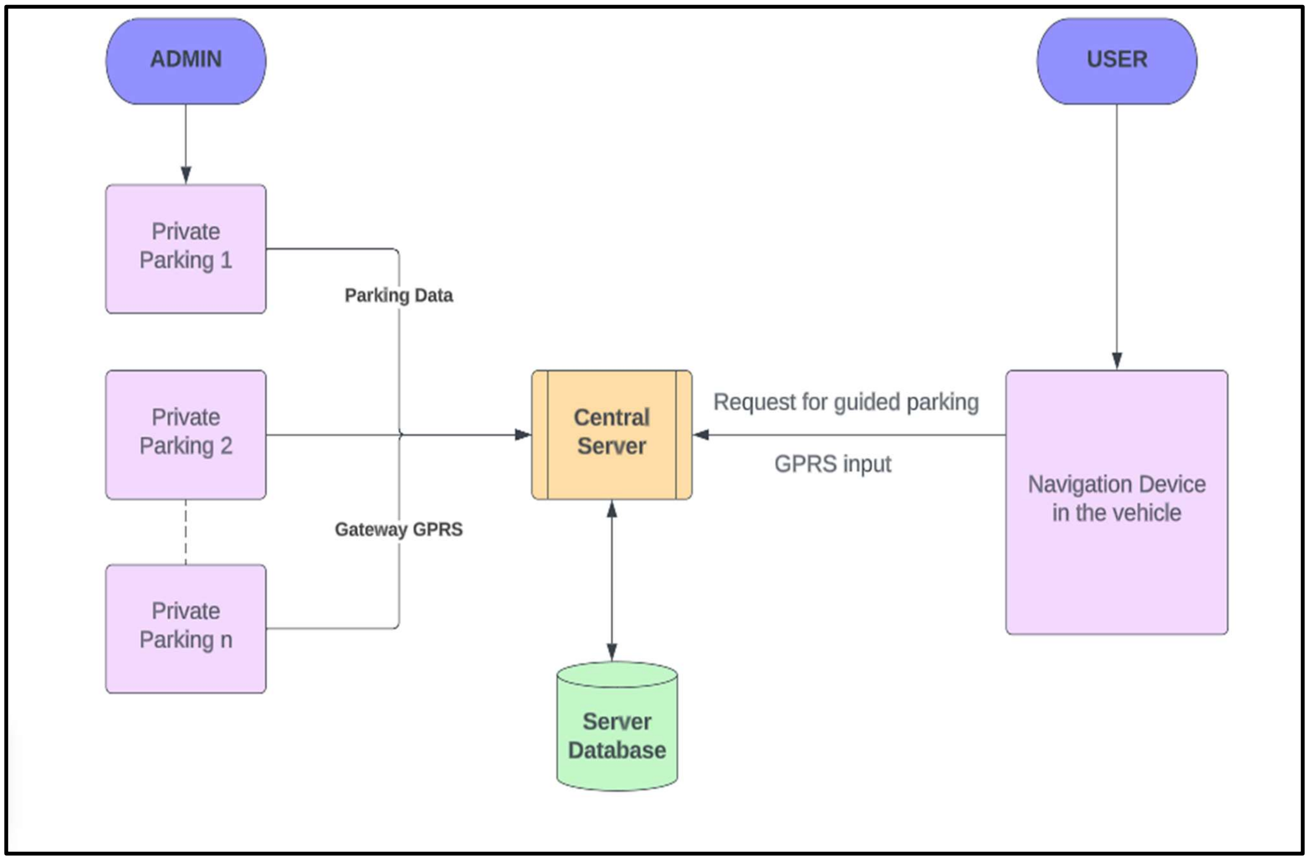


Fig 3.1.1: Architecture Diagram of Server

In Fig 3.1.1, the diagram depicts the flow of information in an online parking system. The central server receives data from multiple parking facilities and stores it in a server database. It also receives vehicle requests along with their GPS positions. As the vehicle moves continuously, its updated position is regularly communicated to the central server. The automated parking guidance system takes into account various parking options in the city. Whenever a request is received, it is forwarded to the central server. The central server, in collaboration with the server database, suggests a parking spot for the requesting vehicle. The driver can either reserve the suggested parking or decline it. If the driver declines, they can request another suggestion and a different parking option will be provided.

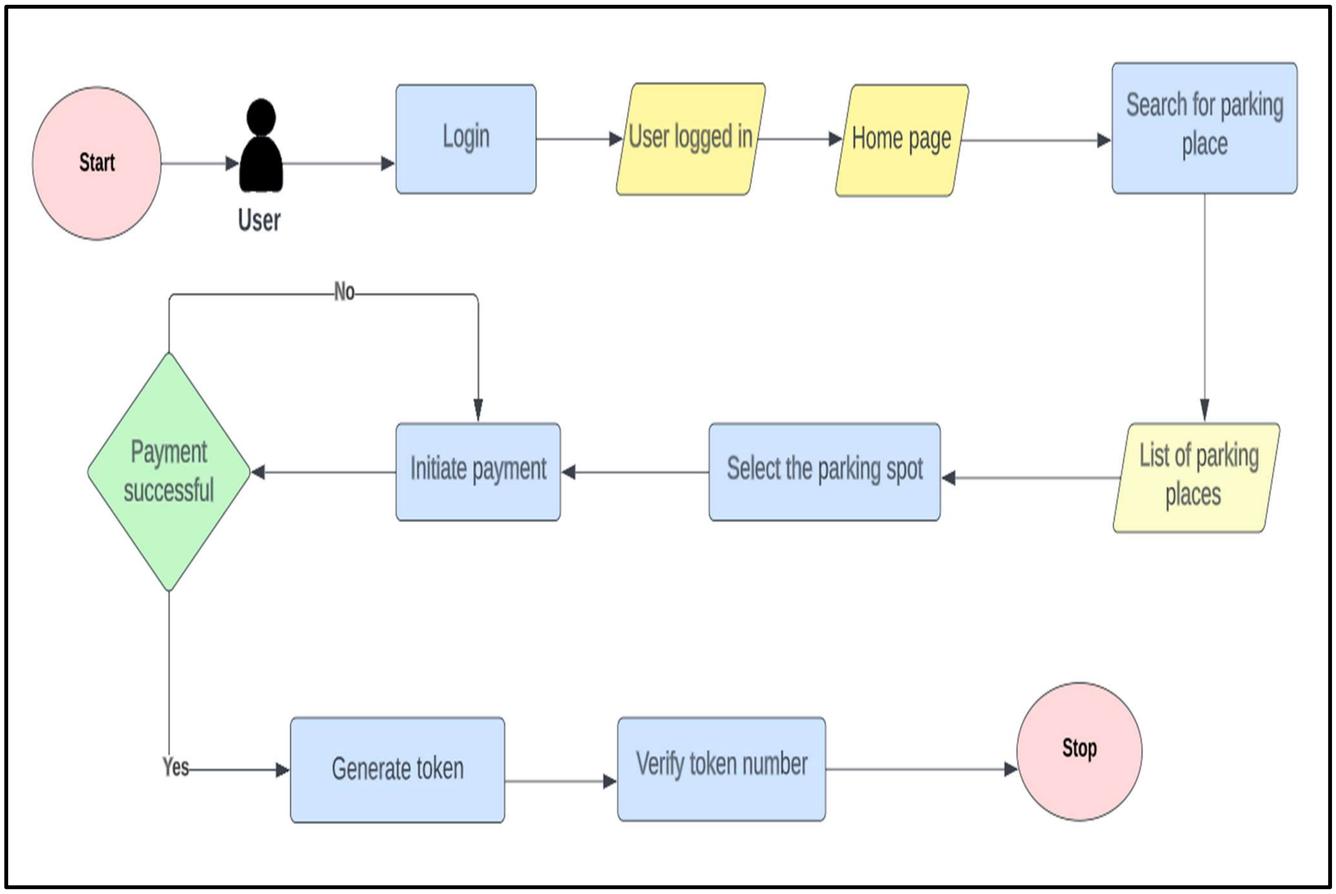


Fig 3.1.2: Architecture Diagram

In Fig 3.1.2, When users open the mobile application, they will see a login screen where they can enter their credentials to access their accounts. If they don't have an account, they can create one. Once logged in, users will be able to search for available parking spaces nearby or in a specific location. They can filter the search results by distance, price, or other criteria to find the perfect parking spot for their needs. When they find a parking spot they like, they can select it and proceed to the checkout page where they can confirm the details of their reservation, such as the date and time, duration, and price. They can then proceed to pay for their reservation using a secure payment gateway integrated into the mobile application. After the payment is processed, users will receive a confirmation of their reservation via email and/or SMS which contains the token number. They can also view their reservation details in their account section within the mobile application.

3.2 Use-Case Diagram

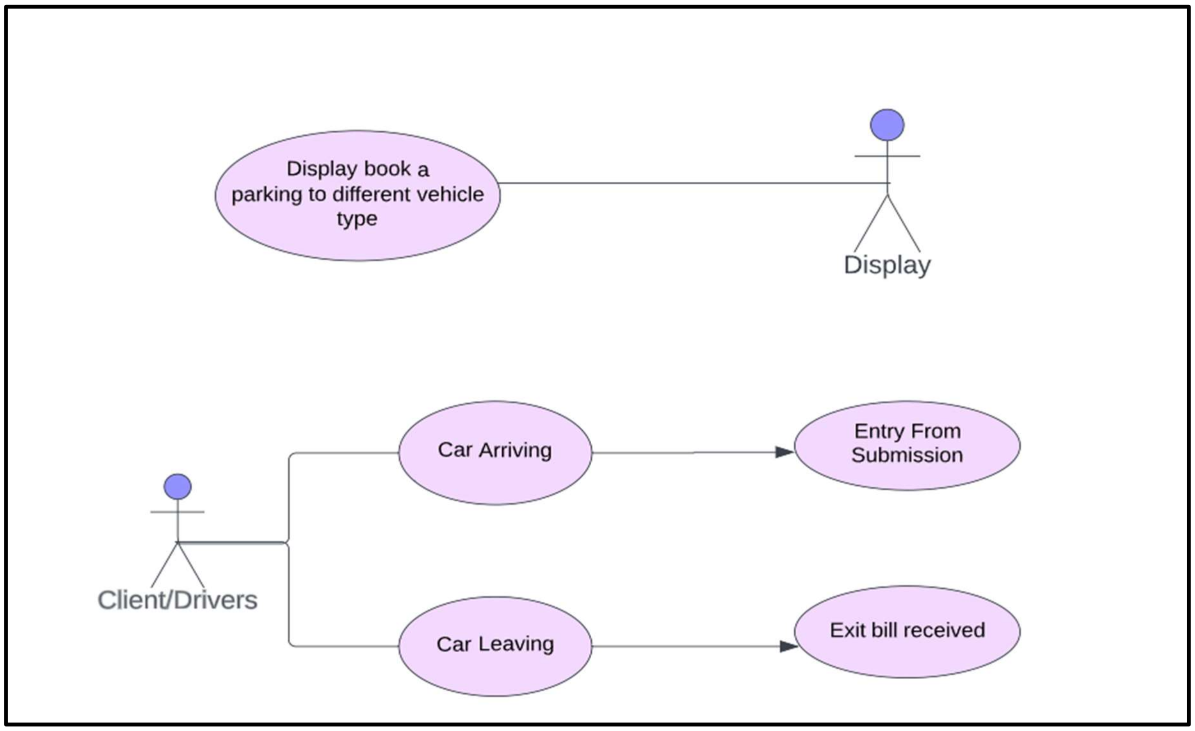


Fig 3.2.1: User Use-Case Diagram

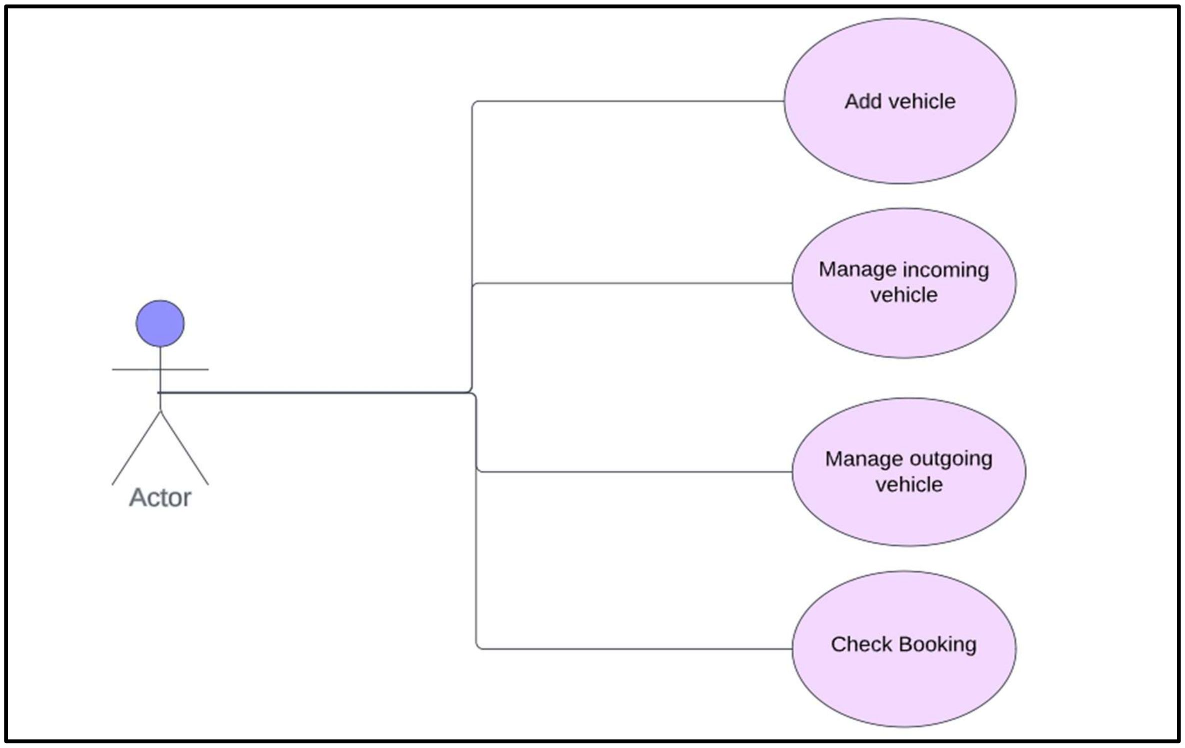


Fig 3.2.2: Admin Use-Case Diagram

In above use case diagrams users and admin will be able to retrieve and manage parking spots and check availability.

3.3 Sequence Diagram

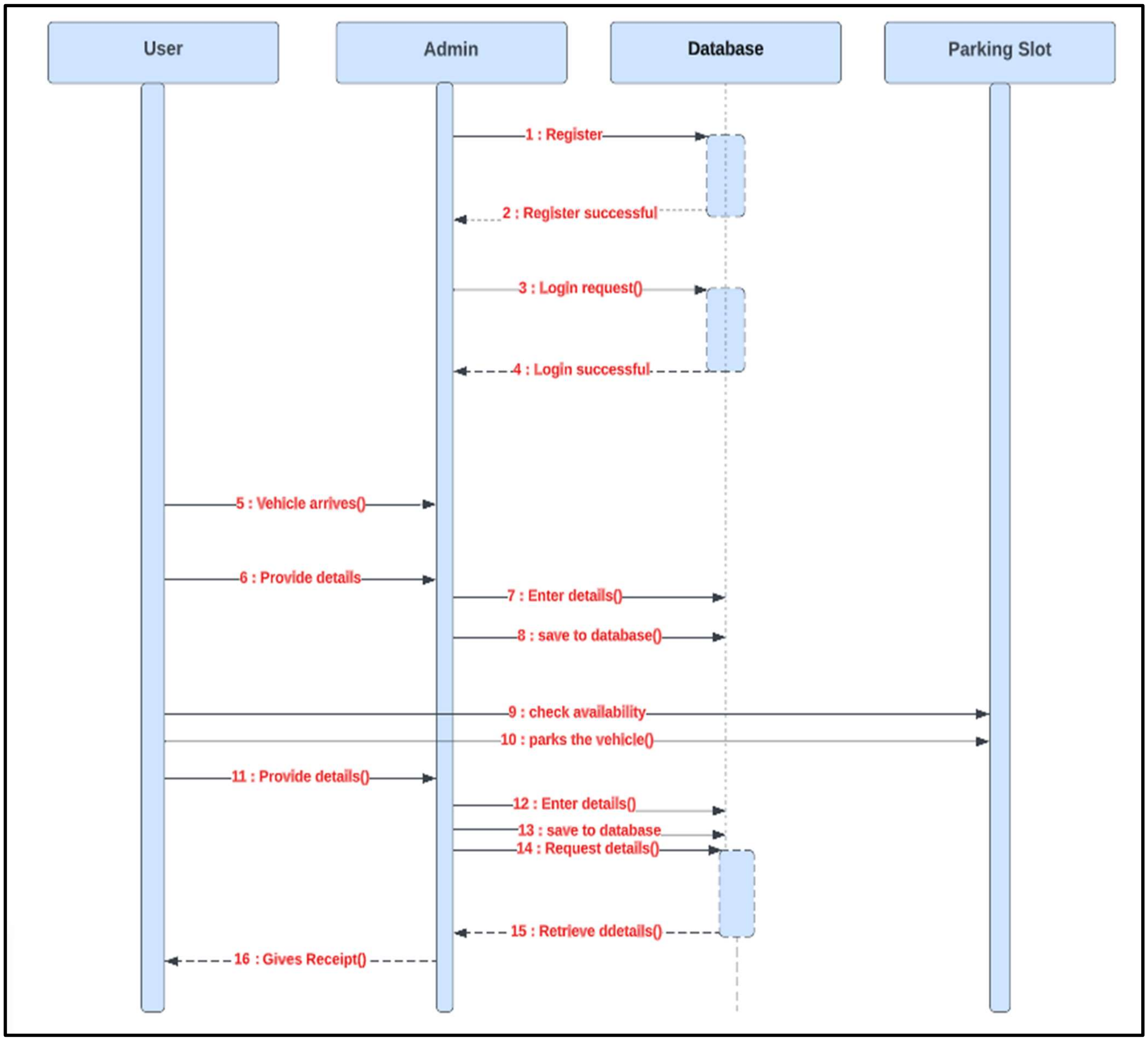


Fig 3.3: Sequence Diagram

This diagram will show the sequence in which the users and admin will be redirected once logged into the app. Once they are logged in, they can view, cancel and book slots and manage online parking system.

3.4 Activity Diagram

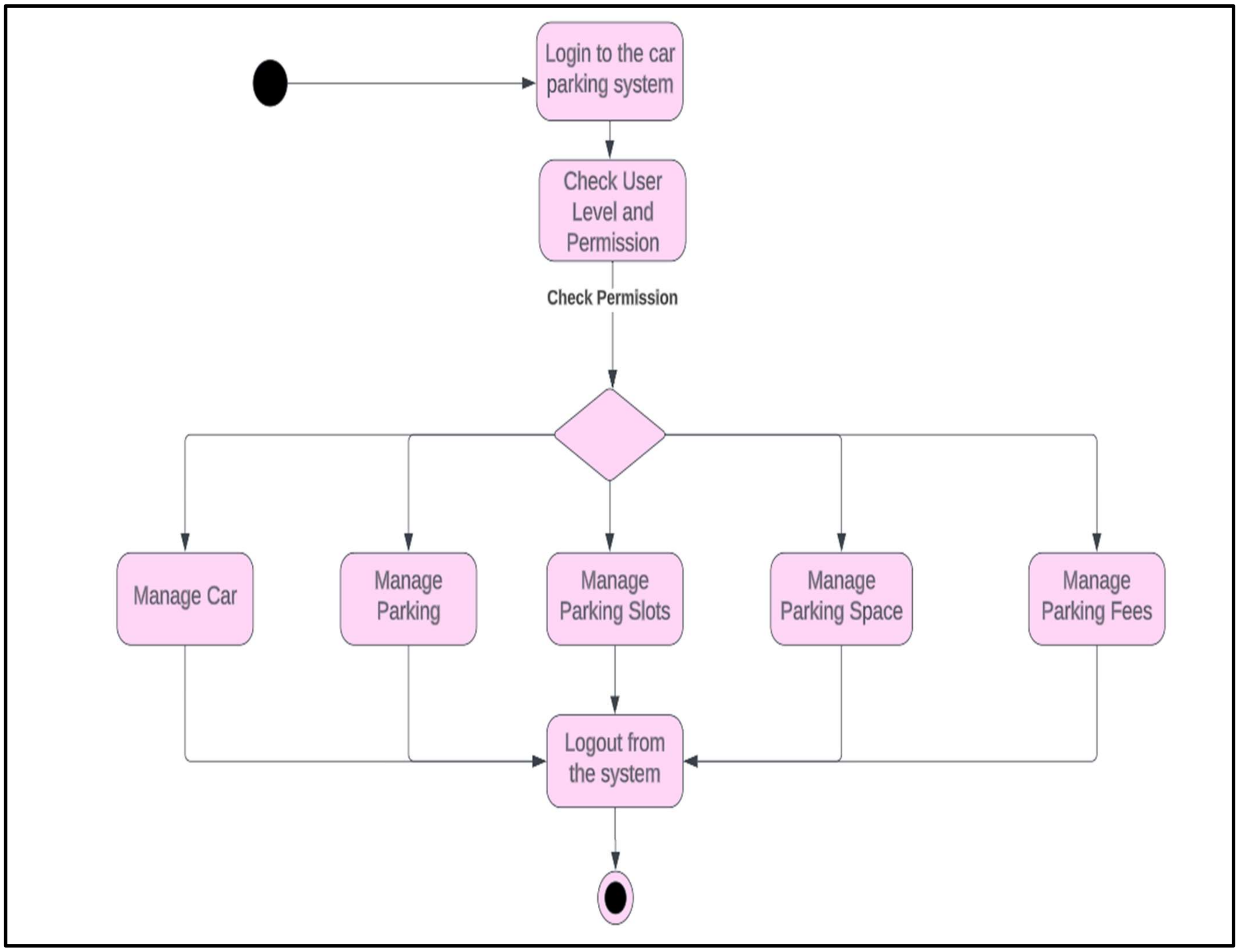


Fig 3.4.1: User Activity Diagram

An activity Diagram is the graphical flow or representation of stepwise action or activity used to define the processes or activity of the models. It is a flowchart that helps to represent the flow from one activity to another the activities are the operation of the models or systems.

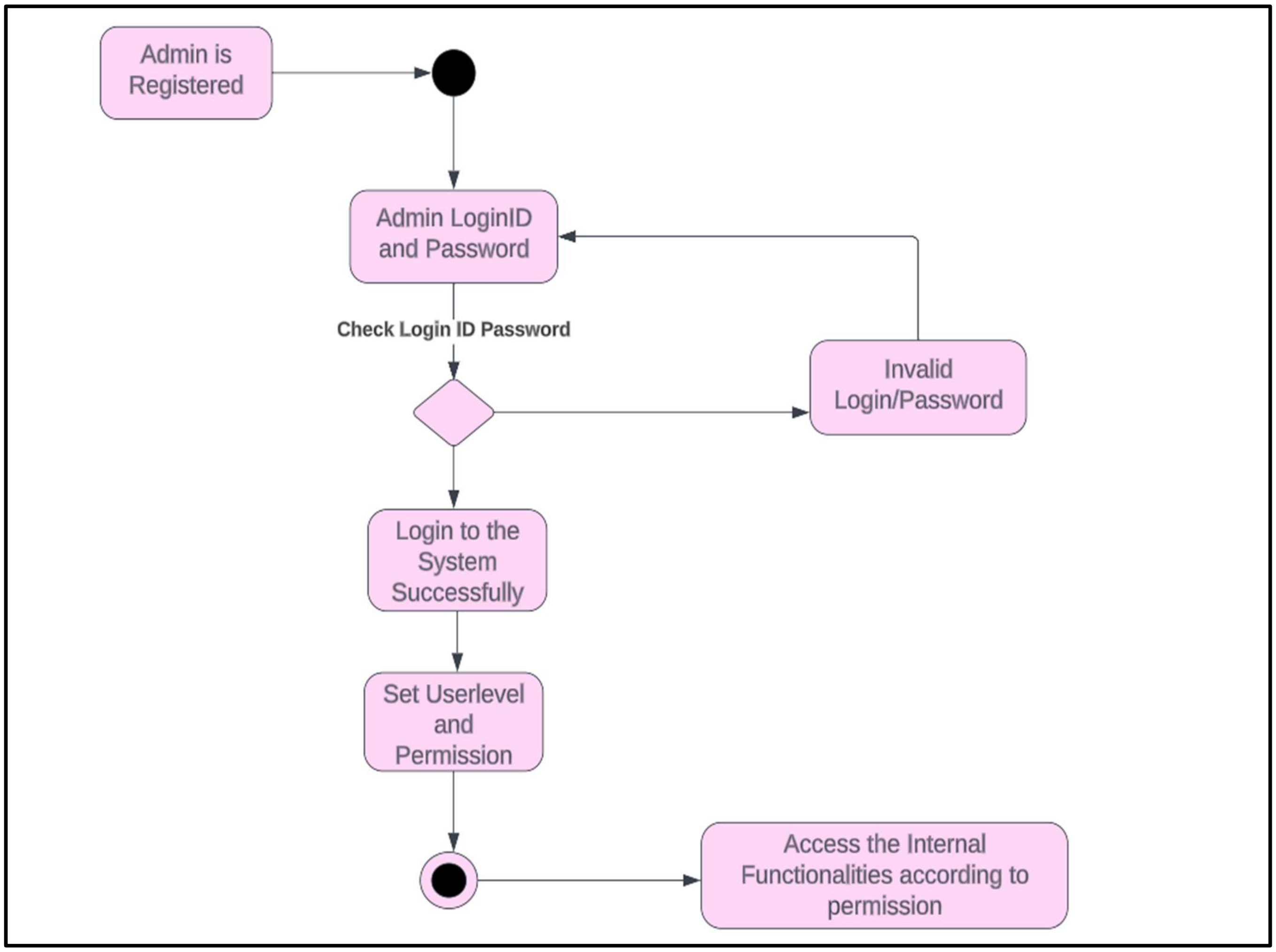

Admin is Registered
Admin LoginID and Password
Check Login ID Password
Invalid Login/Password
Login to the System Successfully
Set Userlevel and Permission
Access the Internal Functionalities according to permission

Fig 3.4.2: Admin Activity Diagram

3.5 Class Diagram

Class Diagram describes the structure of an online parking system classes, their attributes, operations (or methods), and the relationship among objects.

The main classes of the online parking management system are User, Admin, Payment details, Vehicle, and Parking space.

Each class has its own methods.

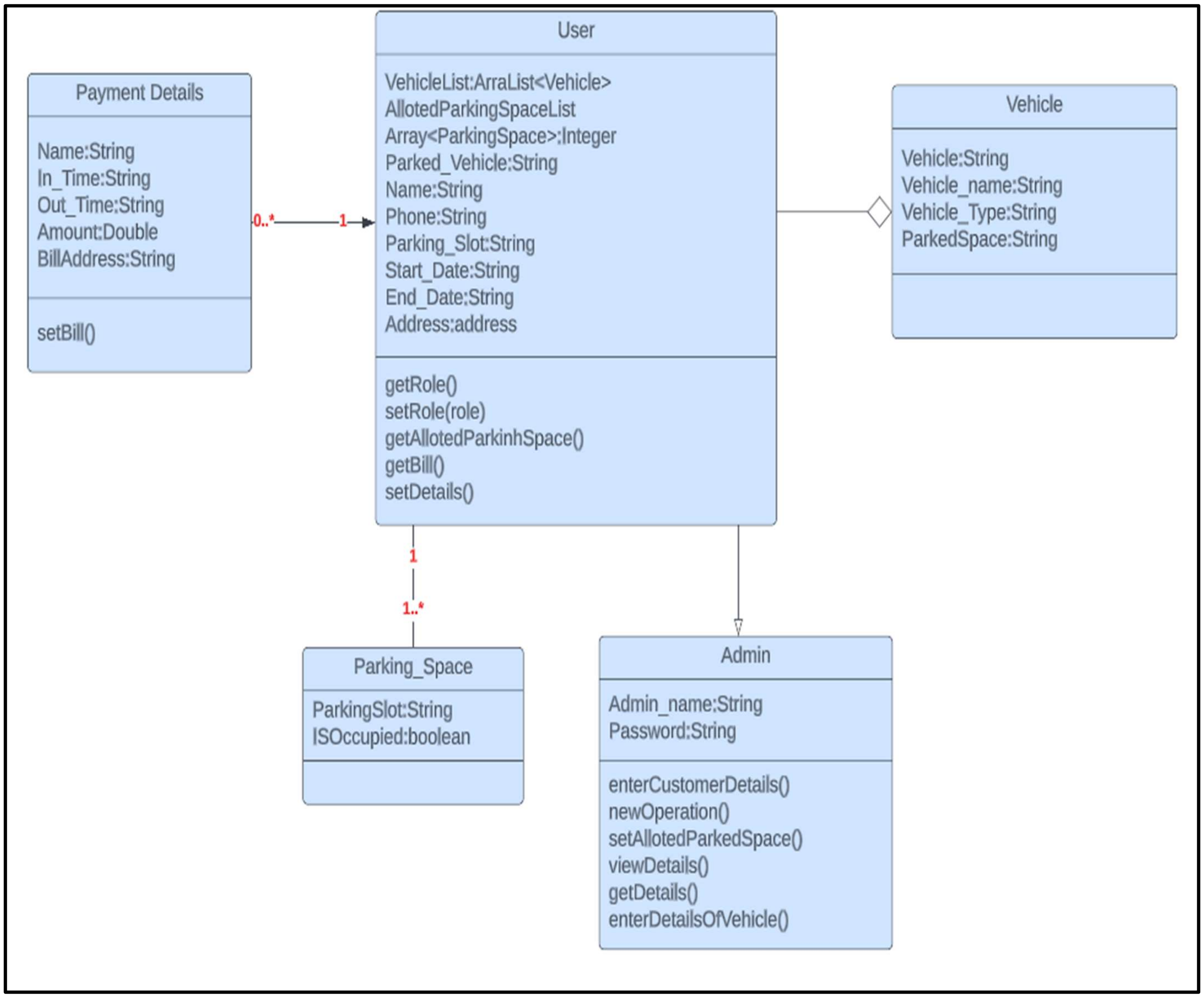


Fig 3.5: Class Diagram

CHAPTER – 4

TOOL DESCRIPTION

4.1 Hardware Requirements: Description

The hardware requirements for users, admins, and retailers at Spypark are as follows -

- 1 GB of RAM
- 1.6 GHz or faster processor
- Hard disk space: 10 GB or above
- Android device for running the application
- Network adapters for connection to the Internet
- Graphics display resolution

4.2 Software Requirements: Description

The software requirements for Spypark are divided into subsections as follows:

4.2.1 Operating System Requirements

- OS X El Capitan (10.11+)
- Windows 7, 8.0, 8.1, and 10 (32-bit and 64-bit)

4.2.2 Client-side Software Requirements

- Google Chrome
- Android Studio
- Git Bash

4.2.3 Developer libraries

- Android
- Android-SDK
- Firebase

These requirements satisfy the Developers, Users and Admins to access Spypark Application for seamless user experience.

CHAPTER – 5

IMPLEMENTATION

5.1 Description

For easy and quick development of the project, we have divided the project into 5 modules.

Module 1	**Development of Splash Screen, setting up of dependencies, Gradle updates, and libraries as well.**
Module 2	**Development of Home Page, Signup, and Login of Users.**
Module 3	**Development of a dashboard page that displays current bookings of vehicles and parking vacancies.**
Module 4	**Storage and Retrieval of data i.e., parking spots added to/from the database.**
Module 5	**Final display of parked and vacant spaces and current user booking details.**

Module 1 - Development of Splash Screen and setting up of dependencies, Gradle updates, and libraries as well.

In this module, we will start with setting up a splash screen containing the logo of the project. Next, we need to connect the Android studio app to Firebase for further operations as first we are doing login/signup functionality. Authentication has to be enabled in Firebase and backend code has to be written for this functionality.

This may involve downloading and configuring software tools, installing libraries and dependencies, creating project directories, and initializing version control. Taking the time to establish these foundations at the start of a project can greatly reduce headaches and obstacles later on.

Module 2 – Development of Home Page, Signup and Login of Users

In this module, we will start building the home page of the application. On the home page, there will be an option to get started, log in and sign up for the application.

New users have to sign up first by providing certain details such as name, email, password, and phone number. After the users get signed up, they need to log in by providing an email and password. While logging in if the wrong password is entered, a toast message will appear mentioning the wrong password inputted.

Module 3 - Development of a dashboard page that displays current bookings of vehicles and parking vacancies.

In this module, the current parking booking of a user typically shows the details of their reservation, such as the location, date, and time of their parking spot. Once users decide to vacate the parking spot, they can click on end booking, and a summary of the parking time and amount to be paid is displayed. It also includes the vehicle number and vehicle type (2-wheeler, 3-wheeler, 4-wheeler). Users can then enter their vehicle number and choose a parking spot based on their preference. This feature allows users to conveniently select the most suitable parking spot for their vehicle and helps them save time by avoiding spots that are already booked.

Module 4 - Storage and Retrieval of data i.e., parking spots added to/from the database.

In this module, we will display a list of available parking spots located near the user's current location. Users can filter the list based on their preferred criteria such as the desired parking spot, duration of the parking session, and personal convenience. It also provides additional information such as parking rates, amenities, operating hours, and special instructions or regulations. Once a parking area is selected, it provides directions and any necessary parking instructions to the user. This feature saves time and effort for the user in finding a suitable parking spot and helps ensure a hassle-free parking experience.

Module 5 - Final display of parked and vacant spaces and current user booking details.

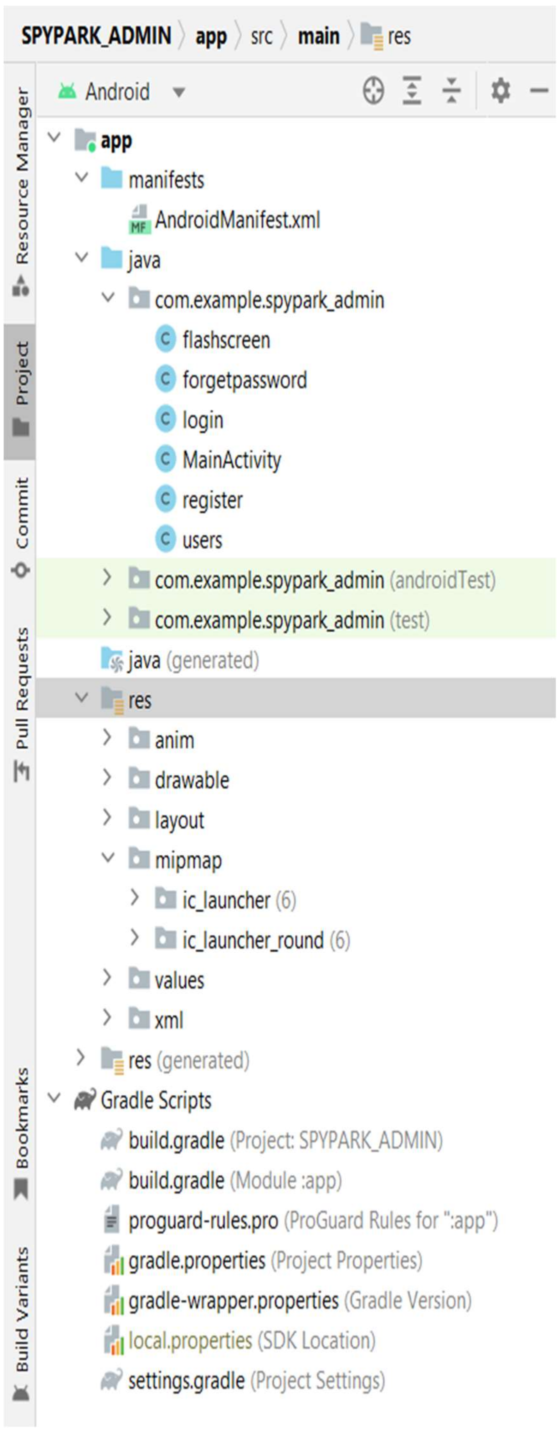
In this module, we will display the number of bikes and cars that are currently booked. Users can then enter their vehicle number and choose a parking spot based on their preference. This feature allows users to conveniently select the most suitable parking spot for their vehicle and helps them save time by avoiding spots that are already booked.

App Components:

They are the essential construction blocks of an Android app. Each component is an introduction point by which bureaucracy or a consumer can record your app. Some parts believe possible choice.

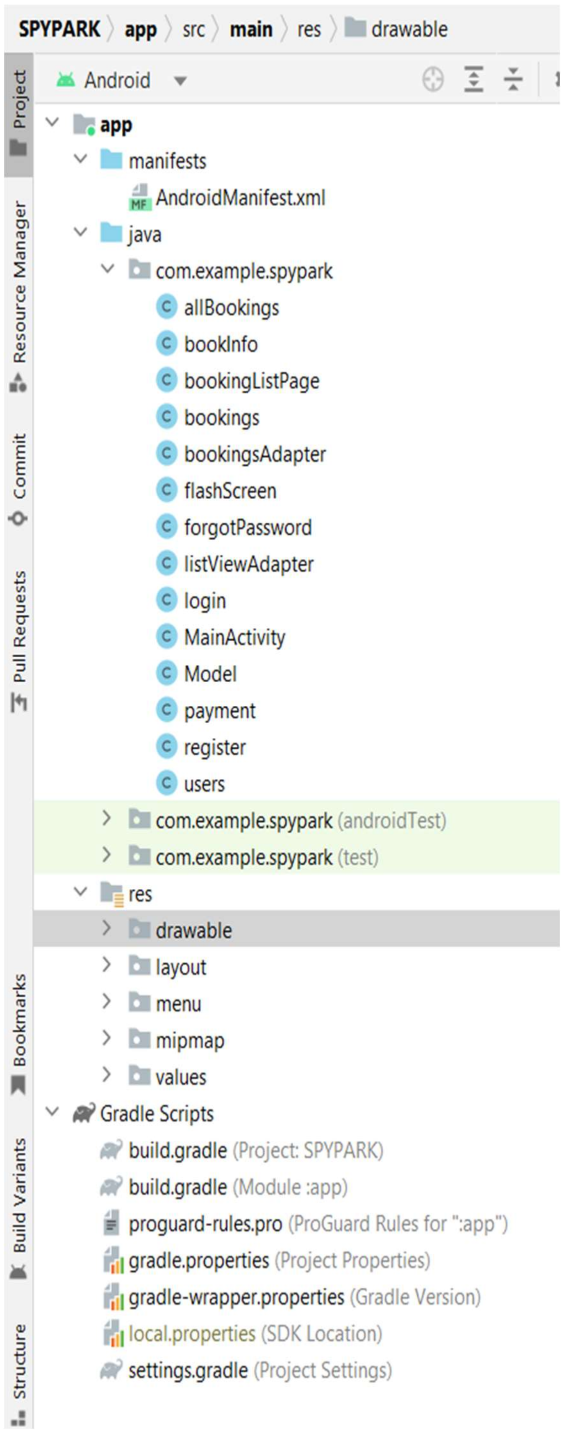
- **Admin-side Login:** The system is supervised by the admin, who manages all registered bookings.
- **User-side login/registration:** Initially, users must register themselves to log into the system.
- **Parking areas:** The system will provide users with parking areas at various locations.
- **Payment gateway integration:** The system integrates with a payment gateway to enable users to make payments online via credit card.
- **Parking slot management:** The system manages and allocates parking slots to users based on their bookings.
- **Real-time updates:** The parking system offers real-time information to users regarding the availability of parking spaces.
- **Alerts and notifications:** The system sends alerts and notifications to users regarding their bookings and any changes in availability.
- **Data analytics and reporting:** The system generates reports and performs data analytics to provide insights on parking usage and trends.
- **Integrated map view:** The system provides an integrated map view of parking areas and available parking slots, making it easier for users to find and book a slot.
- **Security:** The system incorporates security features to protect user data and prevent unauthorized access to the system.
- **Email:** When a user is successful in booking a parking spot at the desired parking spot, the system sends a confirmation email.
- **Feedback:** The system includes a feedback form, through which users can provide feedback to the system.

5.2 Codebase structure breakdown of Spypark Admin and User



```
SPYPARK_ADMIN 〉 app 〉 src 〉 main 〉 res

app
  manifests
    AndroidManifest.xml
  java
    com.example.spypark_admin
      flashscreen
      forgetpassword
      login
      MainActivity
      register
      users
    com.example.spypark_admin (androidTest)
    com.example.spypark_admin (test)
  java (generated)
  res
    anim
    drawable
    layout
    mipmap
      ic_launcher (6)
      ic_launcher_round (6)
    values
    xml
  res (generated)
Gradle Scripts
  build.gradle (Project: SPYPARK_ADMIN)
  build.gradle (Module :app)
  proguard-rules.pro (ProGuard Rules for ":app")
  gradle.properties (Project Properties)
  gradle-wrapper.properties (Gradle Version)
  local.properties (SDK Location)
  settings.gradle (Project Settings)
```



```
SPYPARK 〉 app 〉 src 〉 main 〉 res 〉 drawable

app
  manifests
    AndroidManifest.xml
  java
    com.example.spypark
      allBookings
      bookInfo
      bookingListPage
      bookings
      bookingsAdapter
      flashScreen
      forgotPassword
      listViewAdapter
      login
      MainActivity
      Model
      payment
      register
      users
    com.example.spypark (androidTest)
    com.example.spypark (test)
  res
    drawable
    layout
    menu
    mipmap
    values
Gradle Scripts
  build.gradle (Project: SPYPARK)
  build.gradle (Module :app)
  proguard-rules.pro (ProGuard Rules for ":app")
  gradle.properties (Project Properties)
  gradle-wrapper.properties (Gradle Version)
  local.properties (SDK Location)
  settings.gradle (Project Settings)
```

The Codebase of Spypark is divided neatly into modules and sub-modules.

- The. idea contains all the sdk related libraries required for this project.
- The **android** folder contains directories like main, src that manage all the code required for running the application on android devices.
- The **build** is an inbuilt folder that contains the code related to android and Java.
- The **src** folder contains directories like main, src that manage all the code required for running the application on android devices.
- The **lib** folder is the main folder in the project that contains all the files containing code in it. All the working code is stored in the lib files.
- Next in the main folder inside a package all the activities, and adapters used in the project are present.
- Next inside the res folder the layouts associated with the activities, drawable images, menu, mipmaps, and values are present.
- Even the res folder contains the Manifest xml file.
- Gradle holds the configuration files that define various settings for the project, such as database configurations, environment variables, logging configurations, etc.

5.3 Code Snippets of Spypark Application

flashScreen.java

```
package com.example.spypark;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;

import java.util.concurrent.Delayed;

public class flashScreen extends AppCompatActivity {
    Handler h = new Handler();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_flash_screen);
        h.postDelayed(new Runnable() {
            @Override
            public void run() {
                Intent intent = new Intent(getApplicationContext(),
MainActivity.class);
                startActivity(intent);
                finish();
            }
        }, 2000);
    }
}
```

login.java

```
package com.example.spypark;

import static android.content.ContentValues.TAG;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.textfield.TextInputEditText;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

import org.jetbrains.annotations.NotNull;

public class login extends AppCompatActivity {
    TextInputEditText editTextEmail,editTextPassword;
    Button buttonLogin;
    TextView forgotPassword,register;
    private FirebaseAuth mAuth;
    ProgressBar progressBar;
    @Override
    public void onStart() {
        super.onStart();
        FirebaseUser currentUser = mAuth.getCurrentUser();
        if(currentUser != null){
            Intent intent = new Intent(getApplicationContext(),
MainActivity.class);
            startActivity(intent);
            finish();
        }
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        editTextEmail= findViewById(R.id.login_email);
        editTextPassword = findViewById(R.id.login_password);
        buttonLogin = findViewById(R.id.login_button);
        register= findViewById(R.id.navigate_to_signUp);
        forgotPassword= findViewById(R.id.forgotButton);
        mAuth = FirebaseAuth.getInstance();
        progressBar=findViewById(R.id.login_progress);

        forgotPassword.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(getApplicationContext(),
forgotPassword.class);
                startActivity(intent);
            }
        });

        buttonLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

            }
        });

        register.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(getApplicationContext(),
register.class);
                startActivity(intent);
                finish();
            }
        });
        buttonLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String email, password;

                email= editTextEmail.getText().toString();
                password= editTextPassword.getText().toString();
                progressBar.setVisibility(View.VISIBLE);

                if(TextUtils.isEmpty(email)){
                    Toast.makeText(login.this, "Enter Email",
Toast.LENGTH_SHORT).show();
                    progressBar.setVisibility(View.GONE);
                    return;
                }
                if(TextUtils.isEmpty(password)){
                    Toast.makeText(login.this, "Enter Password",
Toast.LENGTH_SHORT).show();
                    progressBar.setVisibility(View.GONE);
                    return;
                }

                mAuth.signInWithEmailAndPassword(email, password)
                        .addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
                            @Override
                            public void onComplete(@NotNull Task<AuthResult>
task) {
                                if (task.isSuccessful()) {
                                    // Sign in success, update UI with the
signed-in user's information
                                    progressBar.setVisibility(View.GONE);
                                    Toast.makeText(login.this,
"Authentication Success.",Toast.LENGTH_SHORT).show();
                                    FirebaseUser user =
mAuth.getCurrentUser();
                                    Intent intent = new
Intent(getApplicationContext(), MainActivity.class);
                                    startActivity(intent);
                                    finish();
                                } else {
                                    // If sign in fails, display a message to
the user.
                                    Log.w(TAG, "signInWithEmail:failure",
task.getException());
                                    Toast.makeText(login.this,
"Authentication failed.",
                                            Toast.LENGTH_SHORT).show();
                                    progressBar.setVisibility(View.GONE);

                                }
                            }
                        });
            }
        });


    }
}
```

Users.java

```java
package com.example.spypark;

public class users {
    String name, email, phone, password;

    public users() {
    }

    public users(String name, String email, String phone, String password) {
        this.name = name;
        this.email = email;
        this.phone = phone;
        this.password = password;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

register.java

```
package com.example.spypark;

import static android.content.ContentValues.TAG;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import com.example.spypark.databinding.ActivityMainBinding;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.textfield.TextInputEditText;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

import org.jetbrains.annotations.NotNull;

import java.nio.file.Path;

public class register extends AppCompatActivity {
    TextInputEditText editTextName, editTextEmail, editTextPhone,
editTextPassword;
    Button buttonRegister;
    TextView buttonLogin;
    private FirebaseAuth mAuth;
    ProgressBar progressBar;
    ActivityMainBinding binding;
    FirebaseDatabase db;
    DatabaseReference reference;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(R.layout.activity_register);
        editTextName = findViewById(R.id.Name);
        editTextEmail= findViewById(R.id.register_email);
        editTextPhone= findViewById(R.id.phone);
        editTextPassword = findViewById(R.id.register_password);
        buttonRegister = findViewById(R.id.register_button);
        buttonLogin= findViewById(R.id.navigate_to_login);
        mAuth = FirebaseAuth.getInstance();
        progressBar=findViewById(R.id.register_progress);
        buttonRegister.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String email, phone, name, password;
                name= editTextName.getText().toString();
                email= editTextEmail.getText().toString();
                phone= editTextPhone.getText().toString();
                password= editTextPassword.getText().toString();
                progressBar.setVisibility(View.VISIBLE);

                if(TextUtils.isEmpty(name)){
                    Toast.makeText(register.this, "Enter Name",
Toast.LENGTH_SHORT).show();
                    return;
                }
                if(TextUtils.isEmpty(email)){
                    Toast.makeText(register.this, "Enter Email",
Toast.LENGTH_SHORT).show();
                    return;
                }
                if(TextUtils.isEmpty(phone)){
                    Toast.makeText(register.this, "Enter Phone number",
Toast.LENGTH_SHORT).show();
                    return;
                }
                if(TextUtils.isEmpty(password)){
                    Toast.makeText(register.this, "Enter Password",
Toast.LENGTH_SHORT).show();
                    return;
                }

                mAuth.createUserWithEmailAndPassword(email,password)
                        .addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
                            @Override
                            public void onComplete(@NotNull Task<AuthResult>
task) {
                                progressBar.setVisibility(View.GONE);
                                if (task.isSuccessful()) {
                                    // Sign in success, update UI with the
signed-in user's information
                                    FirebaseUser user =
mAuth.getCurrentUser();
                                    Toast.makeText(register.this, "User
Registered.",Toast.LENGTH_SHORT).show();
                                    users users = new users(name, email,
phone, password);
                                    db = FirebaseDatabase.getInstance();
                                    reference = db.getReference("Users");

reference.child(name).setValue(users).addOnCompleteListener(new
OnCompleteListener<Void>() {
                                        @Override
                                        public void onComplete(Task<Void>
task) {
```

```
                                            Intent intent = new
Intent(getApplicationContext(), MainActivity.class);
                                            startActivity(intent);
                                            finish();
                                            editTextName.setText("");
                                            editTextEmail.setText("");
                                            editTextPhone.setText("");
                                            editTextPassword.setText("");

                                        }
                                    });

                                } else {
                                    // If sign in fails, display a message to
the user.
                                    Toast.makeText(register.this,
"Authentication failed.",
                                            Toast.LENGTH_SHORT).show();
                                }
                            }
                        });
            }
        });
        buttonLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(getApplicationContext(),
login.class);
                startActivity(intent);
                finish();
            }
        });
    }

}
```

allBookings.java

```
package com.example.spypark;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.os.Bundle;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.sql.Array;
import java.util.ArrayList;
```

```
public class allBookings extends AppCompatActivity {

    RecyclerView recyclerView;
    DatabaseReference database;
    bookingsAdapter adapter;
    ArrayList<bookings> list;

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

        recyclerView = findViewById(R.id.bookings);
        database = FirebaseDatabase.getInstance().getReference("Bookings");
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        list = new ArrayList<>();
        adapter = new bookingsAdapter(this, list);
        recyclerView.setAdapter(adapter);
        database.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot snapshot) {
                for(DataSnapshot dataSnapshot : snapshot.getChildren()){
                    bookings bookings =
dataSnapshot.getValue(com.example.spypark.bookings.class);
                    list.add(bookings);
                }
                adapter.notifyDataSetChanged();
            }

            @Override
            public void onCancelled(DatabaseError error) {

            }
        });
    }
}
```

bookinfo.java

```
package com.example.spypark;

import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;
import androidx.swiperefreshlayout.widget.SwipeRefreshLayout;

import android.app.Dialog;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.Uri;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.example.spypark.databinding.ActivityMainBinding;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.Query;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.HashMap;

public class bookInfo extends AppCompatActivity {

    DatabaseReference reference;
    TextView bikeTv, carTv,vehchileNumber;
    Button buttonBike, buttonCar, buttonMap;
    ActivityMainBinding binding;
    SwipeRefreshLayout swipeRefreshLayout;
    SharedPreferences sharedPreferences;
    private static final String sharedPreferencesName="myPref";
    private static final String vehchileNumberPref = "0000";
    private static final String vehchileTypePref = "1111";
    private static final String bookStatusPref = "2222";
    private static final String exitBook = "3333";
    private static final String pName = "4444";
    private static final String bookTime = "5555";
    private static final String checkoutTime = "6666";
    private static final String money = "7777";
    private FirebaseDatabase firebaseDatabase;
    private DatabaseReference databaseReference;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(R.layout.activity_book_info);


        bikeTv= findViewById(R.id.bike);
        carTv= findViewById(R.id.car);
        buttonBike = findViewById(R.id.bikeBook);
        buttonCar = findViewById(R.id.carBook);
        swipeRefreshLayout = findViewById(R.id.swipeRefresh);
        vehchileNumber = findViewById(R.id.vechileNumber);
        sharedPreferences =
getSharedPreferences(sharedPreferencesName,MODE_PRIVATE );
        firebaseDatabase = FirebaseDatabase.getInstance();
        databaseReference = firebaseDatabase.getReference();
        buttonMap = findViewById(R.id.map);
        buttonMap.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = getIntent();
                String parkName = intent.getStringExtra("actionbarTitle");
                if (parkName.equals("Phoenix")){
                    gotoUrl("https://goo.gl/maps/qQSQHYDiegz249PB9");
                }
                if (parkName.equals("Orion")){
                    gotoUrl("https://goo.gl/maps/exsHyJGqYdcbPn958");
                }
            }
        });

        ActionBar actionBar = getSupportActionBar();

        Intent intent = getIntent();
        String parkName = intent.getStringExtra("actionbarTitle");

        actionBar.setTitle(parkName);

        readData(parkName);

        swipeRefreshLayout.setOnRefreshListener(new
SwipeRefreshLayout.OnRefreshListener() {
            @Override
            public void onRefresh() {
                Intent intent = new Intent(bookInfo.this, bookInfo.class);
                intent.putExtra("actionbarTitle",parkName);
                startActivity(intent);
                finish();
                swipeRefreshLayout.setRefreshing(false);
            }
        });


        buttonBike.setOnClickListener(new View.OnClickListener() {
            String bike = "bike";

            @Override
            public void onClick(View v) {
                String VN= String.valueOf(vehchileNumber.getText());
                if(TextUtils.isEmpty(VN)){
                    Toast.makeText(bookInfo.this, "Enter Vechile number",
Toast.LENGTH_SHORT).show();
                    return;
                }
                updateData(parkName,bike);
                readData(parkName);
                Calendar calendar = Calendar.getInstance();
                SimpleDateFormat simpleDateFormat = new
SimpleDateFormat("hh:mm:ss a");
                String time= simpleDateFormat.format(calendar.getTime());
                SharedPreferences.Editor editor = sharedPreferences.edit();
                editor.putString(vehchileNumberPref, VN);
                editor.putString(vehchileTypePref, bike);
                editor.putString(bookStatusPref, "done");
                editor.putString(exitBook, "can");
                editor.putString(pName, parkName);
                editor.putString(bookTime, time);
                editor.putString(checkoutTime, "");
                editor.putString(money, "");
                editor.apply();
                HashMap<String,Object> hashMap = new HashMap<>();
                hashMap.put("bookTime",time);
                hashMap.put("parkName",parkName);
                hashMap.put("vechNum",VN);
                hashMap.put("endTime","");
                hashMap.put("money","");
                databaseReference.child("Bookings")
                        .child(VN)
                        .setValue(hashMap)
                        .addOnSuccessListener(new OnSuccessListener<Void>(){
                            @Override
                            public void onSuccess(Void aVoid){
                            }
                        });
                Intent intent = new Intent(bookInfo.this,
MainActivity.class);
                startActivity(intent);
                finish();
            }

        });

        buttonCar.setOnClickListener(new View.OnClickListener() {

            String car = "car";

            @Override

            public void onClick(View view) {
                String VN= String.valueOf(vehchileNumber.getText());
                if(TextUtils.isEmpty(VN)){
                    Toast.makeText(bookInfo.this, "Enter Vechile number",
Toast.LENGTH_SHORT).show();
                    return;
                }
                updateData(parkName,car);
                readData(parkName);
                Calendar calendar = Calendar.getInstance();
                SimpleDateFormat simpleDateFormat = new
SimpleDateFormat("hh:mm:ss a");
                String time= simpleDateFormat.format(calendar.getTime());
                SharedPreferences.Editor editor = sharedPreferences.edit();
                editor.putString(vehchileNumberPref, VN);
                editor.putString(vehchileTypePref, car);
                editor.putString(bookStatusPref, "done");
                editor.putString(exitBook, "can");
                editor.putString(pName, parkName);
                editor.putString(bookTime, time);
                editor.putString(checkoutTime, "");
                editor.putString(money, "");
                editor.apply();
                HashMap<String,Object> hashMap = new HashMap<>();
                hashMap.put("bookTime",time);
                hashMap.put("parkName",parkName);
                hashMap.put("vechNum",VN);
                hashMap.put("endTime","");
                hashMap.put("money","");
                databaseReference.child("Bookings")
                        .child(VN)
                        .setValue(hashMap)
                        .addOnSuccessListener(new OnSuccessListener<Void>(){
                            @Override
                            public void onSuccess(Void aVoid){
                                Toast.makeText(bookInfo.this, "Booking
Confirmed", Toast.LENGTH_SHORT).show();
                            }
                        });
                Intent intent = new Intent(bookInfo.this,
MainActivity.class);
                startActivity(intent);
                finish();
            }
        });

    }

    private void gotoUrl(String s) {
        Uri uri = Uri.parse(s);
        startActivity(new Intent(Intent.ACTION_VIEW, uri));
    }

    private void updateData(String parkName, String object) {

        String bikeCount =  String.valueOf(bikeTv.getText());

        String carCount =  String.valueOf(carTv.getText());
        int cCount= Integer.valueOf(carCount) + 1;
        int bCount= Integer.valueOf(bikeCount) + 1;


        HashMap parkData = new HashMap();

        if (object=="bike")
        {
            parkData.put(object, String.valueOf(bCount));
        }
        if (object=="car")
        {
            parkData.put(object, String.valueOf(cCount));
        }
        reference = FirebaseDatabase.getInstance().getReference("Parkings");

reference.child(parkName).updateChildren(parkData).addOnCompleteListener(new
OnCompleteListener() {
            @Override
            public void onComplete(Task task) {
                if(task.isSuccessful()){
                    if (object=="bike")
                    {
                        bikeTv.setText(String.valueOf(bCount));
                    }
                    if (object=="car")
                    {
                        carTv.setText(String.valueOf(cCount));
                    }
                }
                else{
                    Toast.makeText(bookInfo.this, "Failed to book",
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }

    private void readData(String parkName) {
        reference = FirebaseDatabase.getInstance().getReference("Parkings");
        reference.child(parkName).get().addOnCompleteListener(new
OnCompleteListener<DataSnapshot>() {
            @Override
            public void onComplete(Task<DataSnapshot> task) {
                if(task.isSuccessful()){
                    if(task.getResult().exists()){
                        DataSnapshot dataSnapshot = task.getResult();
                        String bike_count =
String.valueOf(dataSnapshot.child("bike").getValue());
                        String car_count =
String.valueOf(dataSnapshot.child("car").getValue());
                        int bcount = Integer.valueOf(bike_count);
                        int ccount = Integer.valueOf(car_count);
                        if(parkName.equals("Phoenix") && bcount>=120){
                            bikeTv.setText(bike_count);
                            buttonBike.setEnabled(false);
                        }
                        else{
                            bikeTv.setText(bike_count);
                        }
                        if(parkName.equals("Phoenix") && ccount>=80){
                            carTv.setText(car_count);
                            buttonCar.setEnabled(false);
                        }
                        else{
                            carTv.setText(car_count);
                        }
                        if(parkName.equals("Orion") && bcount>=80){
                            bikeTv.setText(bike_count);
                            buttonBike.setEnabled(false);
                        }
                        else{
                            bikeTv.setText(bike_count);
                        }
                        if(parkName.equals("Orion") && ccount>=50){
                            carTv.setText(car_count);
                            buttonCar.setEnabled(false);
                        }
                        else{
                            carTv.setText(car_count);
                        }
                    }
                    else{
                        Toast.makeText(bookInfo.this, "Please wait there is 
some server error 1", Toast.LENGTH_SHORT).show();
                    }
                }
                else{
                    Toast.makeText(bookInfo.this, "Please wait there is some 
server error", Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

bookingListPage.java

```
package com.example.spypark;

import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.Menu;
import android.view.MenuItem;
import android.view.SearchEvent;
import android.view.View;
import android.widget.Adapter;
import android.widget.ListView;
import android.widget.SearchView;
import android.widget.TextView;
import android.widget.SearchView.OnQueryTextListener;

import java.util.ArrayList;

public class bookingListPage extends AppCompatActivity {

    ListView listView;
    listViewAdapter adapter;
    String[] title;
    String[] description;
    int[] icon;
    ArrayList<Model> arrayList = new ArrayList<Model>();

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

        ActionBar actionBar = getSupportActionBar();
        actionBar.setTitle("Parking areas");

        title = new String[]{"Phoenix", "Orion"};
        description = new String[]{"Address", "Address"};
        icon = new int[]{R.drawable.spypark_logo, R.drawable.spypark_logo};

        listView= findViewById(R.id.bookList);

        for (int i =0; i<title.length; i++){
            Model model = new Model(title[i], description[i],icon[i]);
            arrayList.add(model);

        }
        adapter = new listViewAdapter(this, arrayList);
        listView.setAdapter(adapter);
    }

    public boolean onCreateOptionsMenu(Menu menu){

        getMenuInflater().inflate(R.menu.menu, menu);

        MenuItem myActionMenuItem = menu.findItem(R.id.actionSearch);
        SearchView searchView = (SearchView)myActionMenuItem.getActionView();
        searchView.setOnQueryTextListener(new
SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String s) {
                return false;
            }

            @Override
            public boolean onQueryTextChange(String s) {
                if(TextUtils.isEmpty(s)){
                    adapter.filter("");
                    listView.clearTextFilter();
                }
                else{

                    adapter.filter(s);
                }

                return true;
            }
        });

        return true;
    }

}
```

bookingsAdapter.java

```
package com.example.spypark;

import android.content.Context;
import android.telephony.SmsManager;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.recyclerview.widget.RecyclerView;

import com.google.android.material.textfield.TextInputEditText;

import java.net.FileNameMap;
import java.util.ArrayList;

public class bookingsAdapter extends
RecyclerView.Adapter<bookingsAdapter.bookingsViewHolder> {

    Context context;
    ArrayList<bookings> list;

    public bookingsAdapter(Context context, ArrayList<bookings> list) {
        this.context = context;
        this.list = list;
    }

    @Override
    public bookingsViewHolder onCreateViewHolder(ViewGroup parent, int
viewType) {
        View v =
LayoutInflater.from(context).inflate(R.layout.item,parent,false);
        return new bookingsViewHolder(v);
    }

    @Override
    public void onBindViewHolder(bookingsViewHolder holder, int position) {
        bookings bookings = list.get(position);
        holder.vechNum.setText(bookings.getVechNum());
        holder.parkName.setText(bookings.getParkName());
        holder.bookTime.setText(bookings.getBookTime());
        holder.endTime.setText(bookings.getEndTime());
        holder.money.setText(bookings.getMoney());


    }

    @Override
    public int getItemCount() {
        return list.size();
    }

    public static class bookingsViewHolder extends RecyclerView.ViewHolder {

        TextView vechNum, parkName, bookTime, endTime, money;
        public bookingsViewHolder(View itemView) {
            super(itemView);
            vechNum = itemView.findViewById(R.id.vechNum);
            parkName = itemView.findViewById(R.id.parkName);
            bookTime = itemView.findViewById(R.id.bookTime);
            endTime = itemView.findViewById(R.id.endTime);
            money = itemView.findViewById(R.id.money);

        }
    }
}
```

bookings.java

```
package com.example.spypark;

public class bookings {
    String vechNum, parkName, bookTime, endTime, money;

    public String getVechNum() {
        return vechNum;
    }

    public String getParkName() {
        return parkName;
    }

    public String getBookTime() {
        return bookTime;
    }

    public String getEndTime() {
        return endTime;
    }

    public String getMoney() {
        return money;
    }
}
```

listViewAdapter.java

```
package com.example.spypark;

import android.content.Context;
import android.content.Intent;
import android.icu.text.Transliterator;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.TextView;

import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

public class listViewAdapter extends BaseAdapter {

    Context mContext;
    LayoutInflater inflater;
    List<Model> modelList;
    ArrayList<Model> arrayList;

    public listViewAdapter(Context context, List<Model> modelList) {
        mContext = context;
        this.modelList = modelList;
        inflater= LayoutInflater.from(mContext);
        this.arrayList = new ArrayList<Model>();
        this.arrayList.addAll(modelList);

    }

    public class ViewHolder{
        TextView mTitleTv, mDescTv;
        ImageView mIconIv;


    }

    @Override
    public int getCount() {
        return modelList.size();
    }

    @Override
    public Object getItem(int i) {
        return modelList.get(i);
    }

    @Override
    public long getItemId(int i) {
        return i;
    }

    @Override
    public View getView(int position, View view, ViewGroup parent) {

        ViewHolder holder;
        if (view == null){

            holder = new ViewHolder();
            view = inflater.inflate(R.layout.row, null);


            holder.mTitleTv = view.findViewById(R.id.title);
            holder.mIconIv = view.findViewById(R.id.icon);
            holder.mDescTv = view.findViewById(R.id.description);

            view.setTag(holder);

        }
        else{
            holder= (ViewHolder) view.getTag();
        }

        holder.mTitleTv.setText(modelList.get(position).getTitle());
        holder.mDescTv.setText(modelList.get(position).getDesc());
        holder.mIconIv.setImageResource(modelList.get(position).getIcon());


        view.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(modelList.get(position).getTitle().equals("Phoenix")){
                    Intent intent = new Intent(mContext, bookInfo.class);
                    intent.putExtra("actionbarTitle","Phoenix");
                    mContext.startActivity(intent);
                }
                if(modelList.get(position).getTitle().equals("Orion")){
                    Intent intent = new Intent(mContext, bookInfo.class);
                    intent.putExtra("actionbarTitle","Orion");
                    mContext.startActivity(intent);
                }
            }
        });

        return view;
    }

    public void filter(String charText){
        charText = charText.toLowerCase(Locale.getDefault());
        modelList.clear();
        if(charText.length()==0){
            modelList.addAll(arrayList);
        }
        else{
            for(Model model:arrayList){
                if
(model.getTitle().toLowerCase(Locale.getDefault()).contains(charText)){
                    modelList.add(model);
                }
            }
        }
        notifyDataSetChanged();
    }
}
```

Model.java

```
package com.example.spypark;

public class Model {
    String title, desc;
    int icon;

    public Model(String title, String desc, int icon) {
        this.title = title;
        this.desc = desc;
        this.icon = icon;
    }

    public String getTitle() {
        return this.title;
    }

    public String getDesc() {
        return this.desc;
    }

    public int getIcon() {
        return this.icon;
    }

}
```

payment.java

```
package com.example.spypark;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;

public class payment extends AppCompatActivity {
    Handler h = new Handler();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_payment);
        h.postDelayed(new Runnable() {
            @Override
            public void run() {
                Intent intent = new Intent(getApplicationContext(),
MainActivity.class);
                startActivity(intent);
                finish();
            }
        }, 2000);
    }
}
```

MainActivity.java

```
package com.example.spypark;

import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;

import android.content.Intent;
import android.content.SharedPreferences;
import android.net.Uri;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.HashMap;

public class MainActivity extends AppCompatActivity {

    TextView textViewEmail, textViewVehchileNumber, textViewVehchileType,
textViewParkName,bookingTime,checkout,amount;
    CardView amountView;
    Button  navigateToBook, buttonexitPark, buttonMap, allBookings;
    FirebaseAuth auth;
    FirebaseUser user;
    SharedPreferences sharedPreferences;
    DatabaseReference reference;
    private static final String sharedPreferencesName="myPref";
    private static final String vehchileNumberPref = "0000";
    private static final String vehchileTypePref = "1111";
    private static final String bookStatusPref = "2222";
    private static final String exitBook = "3333";
    private static final String pName = "4444";
    private static final String bookTime = "5555";
    private static final String checkoutTime = "6666";
    private static final String money = "7777";
    private FirebaseDatabase firebaseDatabase;
    private DatabaseReference databaseReference;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        textViewEmail = findViewById(R.id.displayEmail);
        navigateToBook = findViewById(R.id.navigateToBook);
        textViewVehchileNumber = findViewById(R.id.vehchileNumber);
        textViewVehchileType = findViewById(R.id.vehchileType);
        textViewParkName = findViewById(R.id.parkName);
        buttonexitPark = findViewById(R.id.exitPark);
        amountView = findViewById(R.id.amountView);
        bookingTime = findViewById(R.id.bookingTime);
        checkout = findViewById(R.id.checkout);
        amount = findViewById(R.id.amount);
        sharedPreferences = getSharedPreferences(sharedPreferencesName,
MODE_PRIVATE);
        firebaseDatabase = FirebaseDatabase.getInstance();
        databaseReference = firebaseDatabase.getReference();
        buttonMap = findViewById(R.id.map);
        allBookings = findViewById(R.id.allBookings);
        buttonMap.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String parkName = sharedPreferences.getString(pName, null);
                if (parkName.equals("Phoenix")){
                    gotoUrl("https://goo.gl/maps/qQSQHYDiegz249PB9");
                }
                if (parkName.equals("Orion")){
                    gotoUrl("https://goo.gl/maps/exsHyJGqYdcbPn958");
                }
            }
        });
        allBookings.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(getApplicationContext(),
allBookings.class);
                startActivity(intent);
            }
        });

        String vNumber = sharedPreferences.getString(vehchileNumberPref,
null);
        String vType = sharedPreferences.getString(vehchileTypePref, null);
        String bStatus = sharedPreferences.getString(bookStatusPref, null);
        String exitStatus = sharedPreferences.getString(exitBook, null);
        String parkName = sharedPreferences.getString(pName, null);
        String timeWhenBooked = sharedPreferences.getString(bookTime, null);
        bookingTime.setText(timeWhenBooked);
        String checkout_Time = sharedPreferences.getString(checkoutTime,
null);
        String moneyPay = sharedPreferences.getString(money, null);
        amount.setText(String.valueOf(moneyPay));
        checkout.setText(checkout_Time);


        if (vNumber != null && vType != null && bStatus != null) {
            textViewVehchileNumber.setText(vNumber);
            textViewVehchileType.setText(vType);
            textViewParkName.setText(parkName);
            if (bStatus.equals("done")) {
                navigateToBook.setEnabled(false);
            } else {
                navigateToBook.setEnabled(true);
            }
            if (exitStatus.equals("can")) {
                buttonexitPark.setEnabled(true);
                buttonMap.setEnabled(true);

            } else {
                buttonexitPark.setEnabled(false);
                buttonMap.setEnabled(false);
            }


        } else {
            Toast.makeText(MainActivity.this, "Unable to get text",
Toast.LENGTH_SHORT).show();
        }


        auth = FirebaseAuth.getInstance();
        user = auth.getCurrentUser();
        if (user == null) {
            Intent intent = new Intent(getApplicationContext(), login.class);
            startActivity(intent);
            finish();
        } else {
            textViewEmail.setText(user.getEmail());
        }
        navigateToBook.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getApplicationContext(),
bookingListPage.class);
                startActivity(intent);
                finish();

            }
        });
        buttonexitPark.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String vType = sharedPreferences.getString(vehchileTypePref,
null);

                updateData(parkName, vType);

                navigateToBook.setEnabled(true);

                Calendar calendar = Calendar.getInstance();
                SimpleDateFormat simpleDateFormat = new
SimpleDateFormat("hh:mm:ss a");
                String timeWhenReleased=
simpleDateFormat.format(calendar.getTime());
                textViewVehchileNumber.setText("");
                textViewVehchileType.setText("");
                textViewParkName.setText("");
                buttonexitPark.setEnabled(false);
                buttonMap.setEnabled(false);
                SharedPreferences.Editor editor = sharedPreferences.edit();
                editor.putString(vehchileNumberPref, "");
                editor.putString(vehchileTypePref, "");
                editor.putString(bookStatusPref, "");
                editor.putString(exitBook, "can't");
                editor.putString(pName, "");
                editor.putString(checkoutTime, timeWhenReleased);
                editor.apply();
                String checkin_Time = sharedPreferences.getString(bookTime,
null);
                String checkout_Time =
sharedPreferences.getString(checkoutTime, null);
                checkout.setText(String.valueOf(checkout_Time));
                int startHour = Integer.valueOf(checkin_Time.substring(0,2));
                int startMin = Integer.valueOf(checkin_Time.substring(3,5));
                int endHour = Integer.valueOf(checkout_Time.substring(0,2));
                int endMin = Integer.valueOf(checkout_Time.substring(3,5));

                int hour= (endHour - startHour)*60;
                int min = endMin  - startMin;
                int amounT = (hour + min)/60;
                SharedPreferences.Editor editor1 = sharedPreferences.edit();

                if(amounT<=1){
                    editor1.putString(money, "30");
                    editor1.apply();
                    amount.setText("30");
                }
                else{
                    int moneyAmount = amounT*30;
                    editor1.putString(money, String.valueOf(moneyAmount));
                    editor1.apply();
                    amount.setText(String.valueOf(moneyAmount));
                }
                String dataMoney = sharedPreferences.getString(money, null);
                HashMap<String,Object> hashMap = new HashMap<>();
                hashMap.put("bookTime",timeWhenBooked);
                hashMap.put("parkName",parkName);
                hashMap.put("vechNum",vNumber);
                hashMap.put("endTime",checkout_Time);
                hashMap.put("money",dataMoney);
                reference =
FirebaseDatabase.getInstance().getReference("Bookings");

reference.child(vNumber).updateChildren(hashMap).addOnCompleteListener(new
OnCompleteListener() {
                    @Override
                    public void onComplete(Task task) {
                    }
                });
                Intent intent = new Intent(MainActivity.this, payment.class);
                startActivity(intent);
                finish();
            }
        });

    }
    private void updateData(String parkName, String object) {
        //read data
        reference = FirebaseDatabase.getInstance().getReference("Parkings");
        reference.child(parkName).get().addOnCompleteListener(new
OnCompleteListener<DataSnapshot>() {
            @Override
            public void onComplete(Task<DataSnapshot> task) {
                if(task.isSuccessful()){
                    if(task.getResult().exists()){
                        DataSnapshot dataSnapshot = task.getResult();
                        String bike_count =
String.valueOf(dataSnapshot.child("bike").getValue());
                        String car_count =
String.valueOf(dataSnapshot.child("car").getValue());
                        int bcount = Integer.valueOf(bike_count)-1;
                        int ccount = Integer.valueOf(car_count)-1;
                        HashMap parkData = new HashMap();

                        if (object=="bike")
                        {
                            parkData.put(object, String.valueOf(bcount));
                        }
                        if (object=="car")
                        {
                            parkData.put(object, String.valueOf(ccount));
                        }
                        reference =
FirebaseDatabase.getInstance().getReference("Parkings");

reference.child(parkName).updateChildren(parkData).addOnCompleteListener(new
OnCompleteListener() {
                            @Override
                            public void onComplete(Task task) {
                                if(task.isSuccessful()){
                                    Toast.makeText(MainActivity.this,
"Booking ended", Toast.LENGTH_SHORT).show();
                                }
                                else{
                                    Toast.makeText(MainActivity.this, "Failed
to exit", Toast.LENGTH_SHORT).show();
                                }
                                }
                            });

                        }
                    }
                }
            });
        }

    public boolean onCreateOptionsMenu(Menu menu) {

        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()){
            case R.id.logout: {
                FirebaseAuth.getInstance().signOut();
                Intent intent = new Intent(getApplicationContext(),
login.class);
                startActivity(intent);
                finish();
            }

        }
        return super.onOptionsItemSelected(item);
    }
    private void gotoUrl(String s) {
        Uri uri = Uri.parse(s);
        startActivity(new Intent(Intent.ACTION_VIEW, uri));
    }
}
```

CHAPTER 6

SYSTEM TESTING AND RESULTS

6.1 System Testing and Results

System testing of an online parking reservation mobile application involves evaluating its functionalities, performance, usability, and reliability. The following are key areas to consider for testing:

1. **Functionality Testing:**
   - ➢ Verify that all essential features of the application are working correctly, such as user registration, login/logout, parking spot search, reservation, cancellation, and payment processing.
   - ➢ Test different scenarios, such as reserving available spots, handling conflicts for overlapping reservations, and displaying accurate information about parking availability.
2. **User Interface Testing**
   - ➢ Ensure that the mobile application's user interface is intuitive, user-friendly, and visually appealing.
   - ➢ Test the navigation flow to ensure smooth transitions between screens and consistent UI elements across different devices and screen sizes.
3. **Performance Testing**
   - ➢ Measure the response time of the application and check if it meets acceptable performance benchmarks.
   - ➢ Test the application's stability and resource usage to ensure it does not crash or consume excessive device resources.
4. **Security Testing**
   - ➢ Verify that user data, including personal information and payment details, is securely transmitted and stored.
   - ➢ Validate that authentication and authorization mechanisms are implemented correctly to prevent unauthorized access.

6.1.1 Splash Screen

6.1.2 Login Page

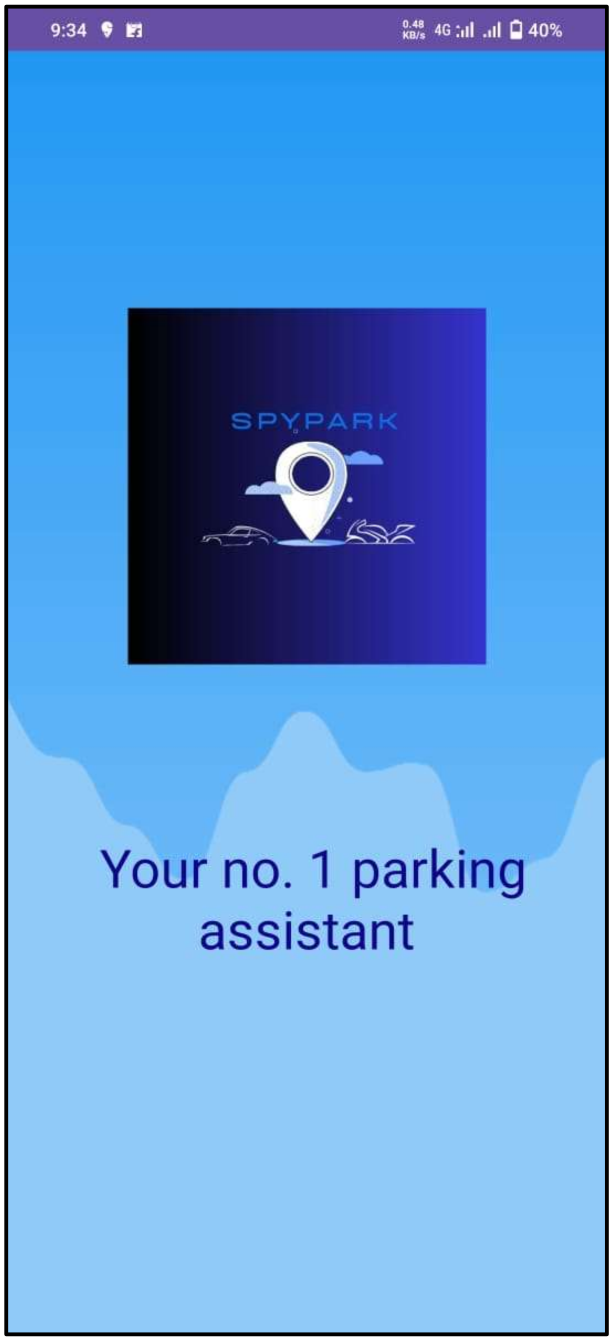


Fig 6.1.1: Splash Screen

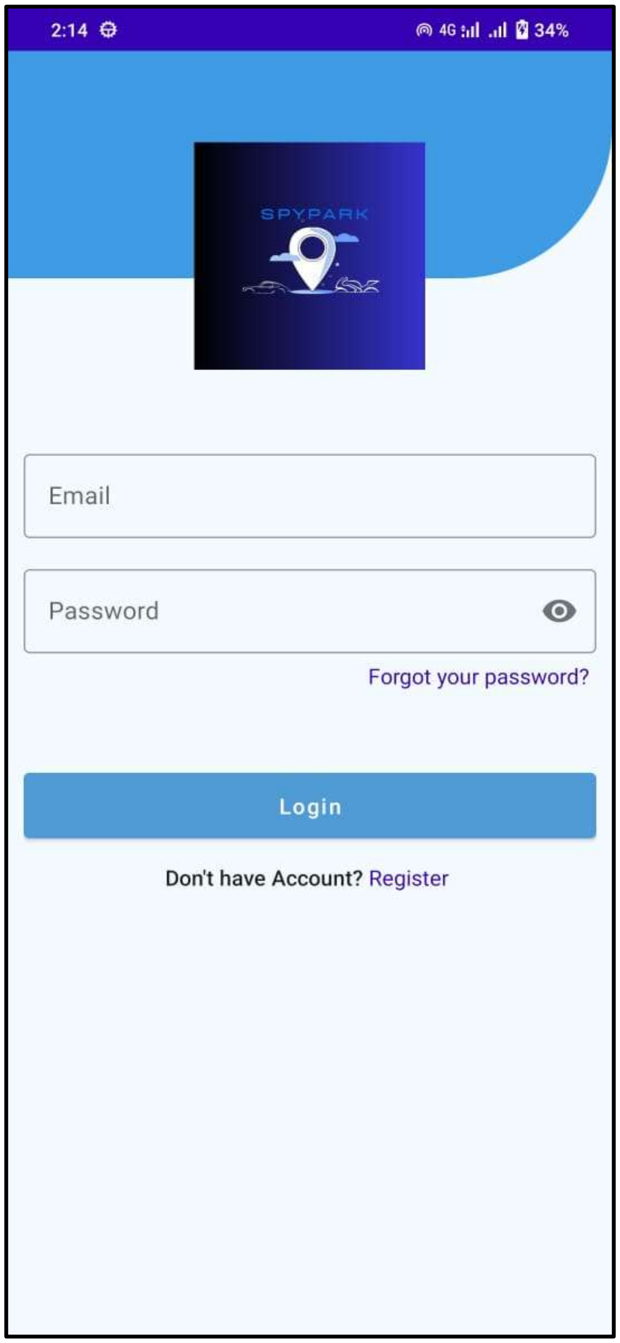


Fig 6.1.2: Login Page

Description of the above images are mentioned in implementation chapter.

6.1.3 Sign Up Page

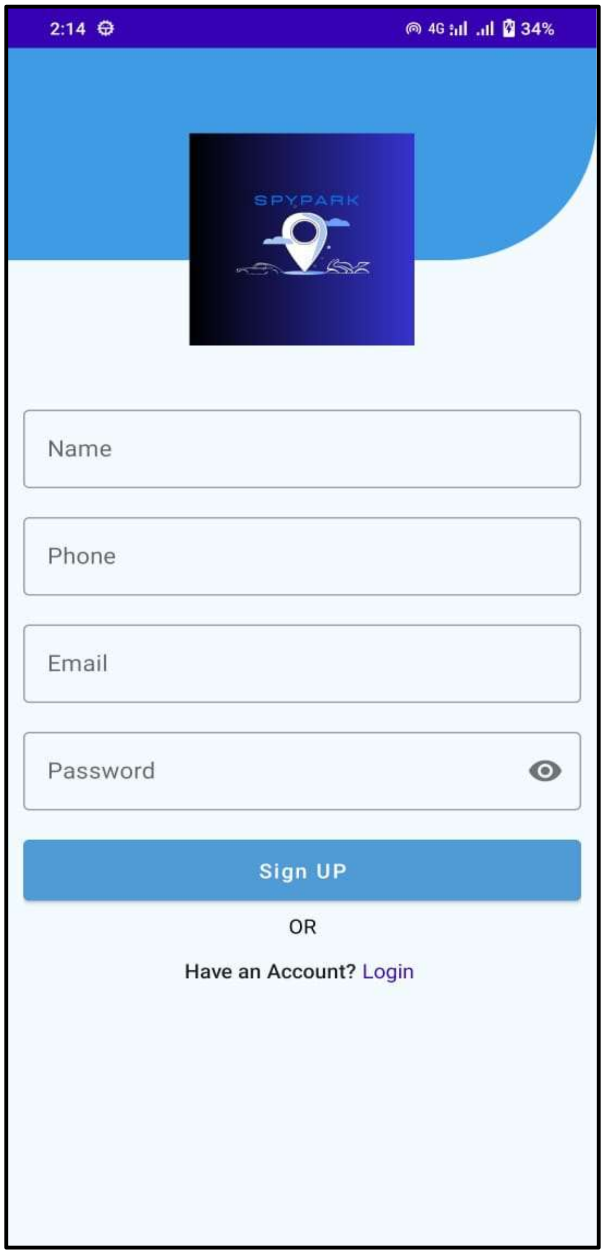


Fig 6.1.3: Sign Up Page

6.1.4 Forgot Password Page

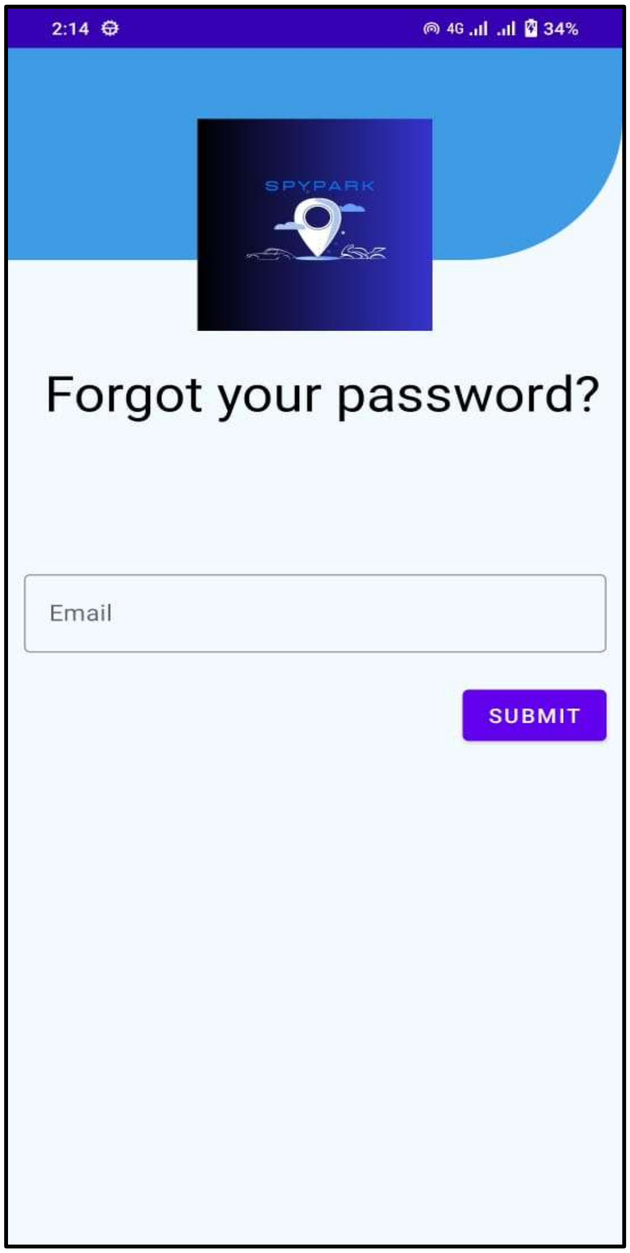
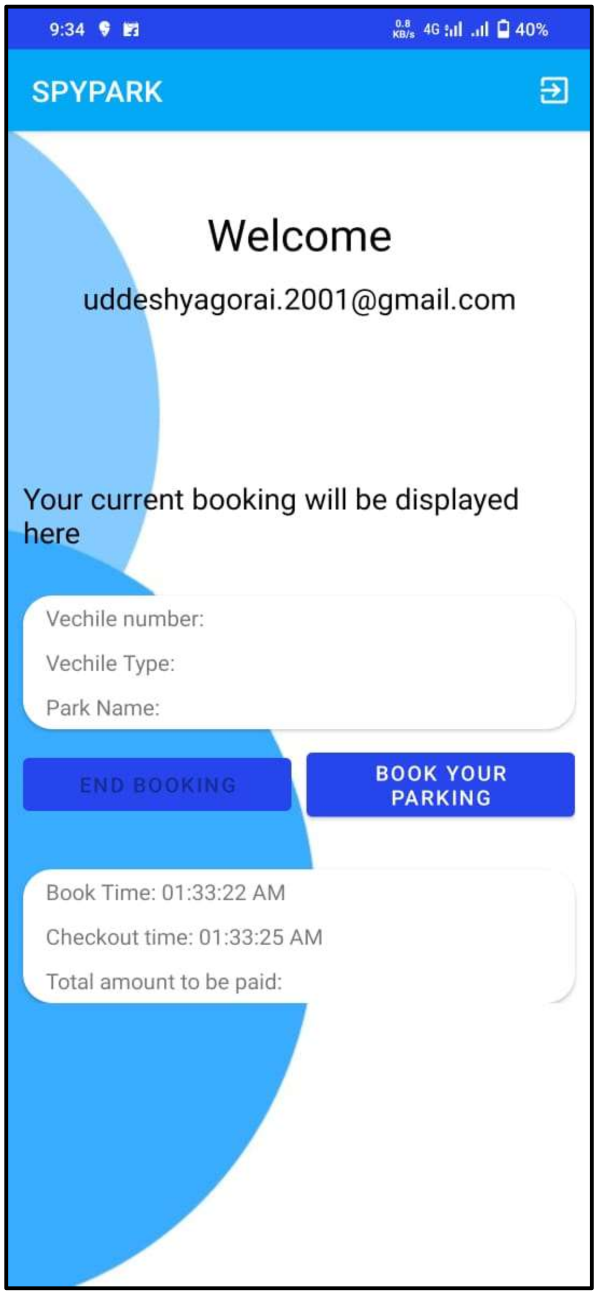


Fig 6.1.4: Forgot Password Page

Description of the above images are mentioned in implementation chapter.

6.1.5 Welcome Page



Welcome

uddeshyagorai.2001@gmail.com

Your current booking will be displayed here

Vechile number:

Vechile Type:

Park Name:

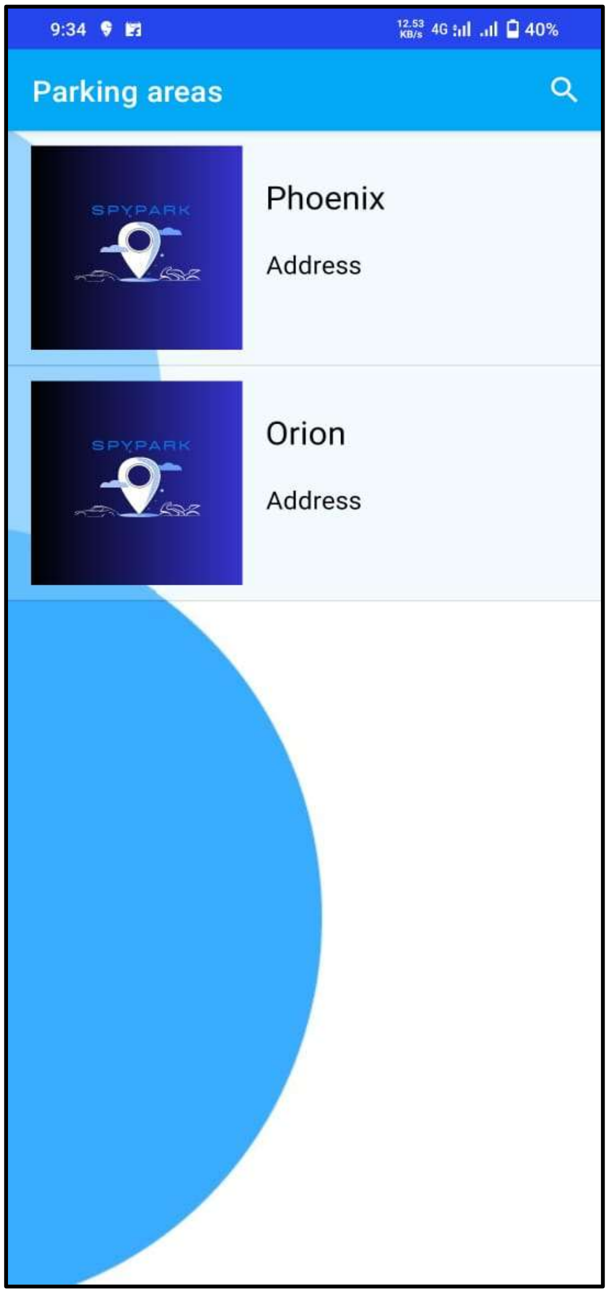
Book Time: 01:33:22 AM

Checkout time: 01:33:25 AM

Total amount to be paid:

Fig 6.1.5: Welcome Page

6.1.6 Parking Areas



Phoenix

Address

Orion

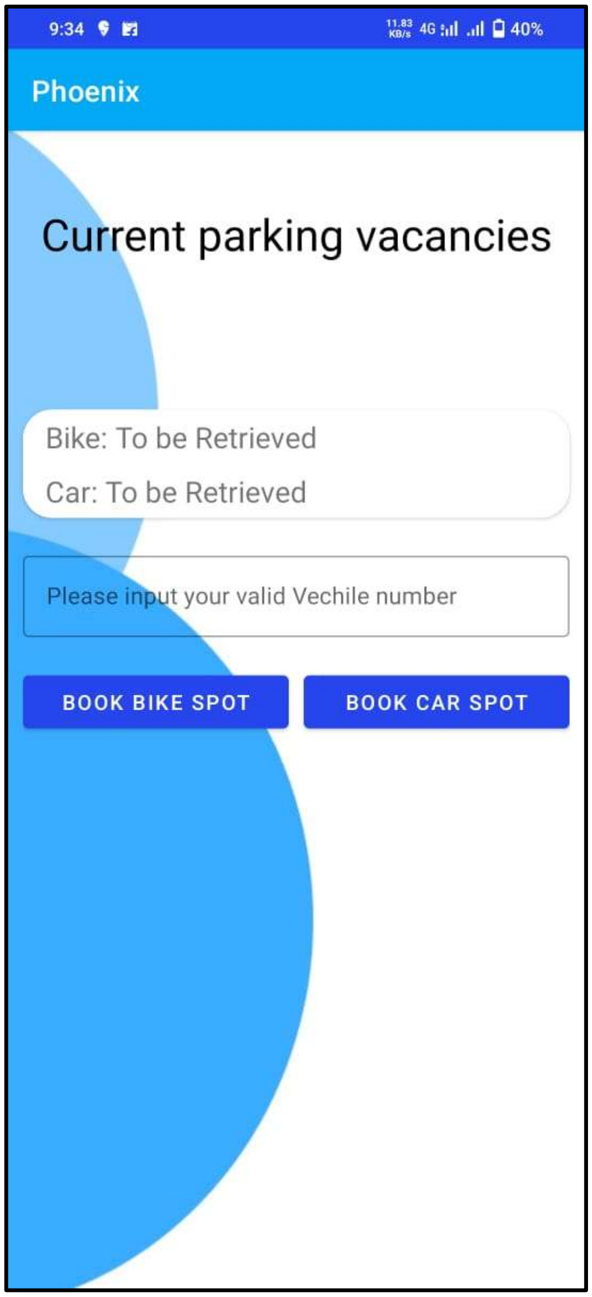
Address

Fig 6.1.6: Parking Areas

Description of the above images are mentioned in implementation chapter.

6.1.7 Parking Area Vacancy

6.1.8 Parking Area Vacancy


9:34 11.83 KB/s 4G 40%

Phoenix

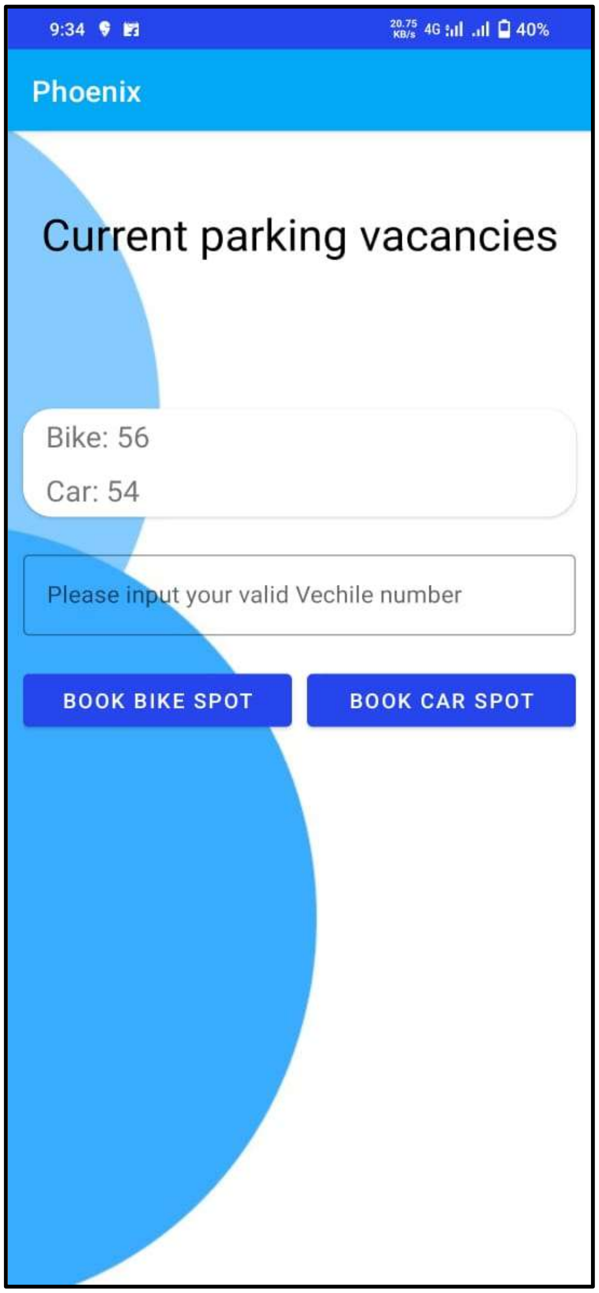
Current parking vacancies

Bike: To be Retrieved

Car: To be Retrieved

Please input your valid Vechile number

Fig 6.1.7: Parking Area Vacancy


9:34 20.75 KB/s 4G 40%

Phoenix

Current parking vacancies

Bike: 56

Car: 54

Please input your valid Vechile number

Fig 6.1.8: Parking Area Vacancy

Description of the above images are mentioned in implementation chapter.

6.1.9 Items Page **6.1.10 All Bookings**

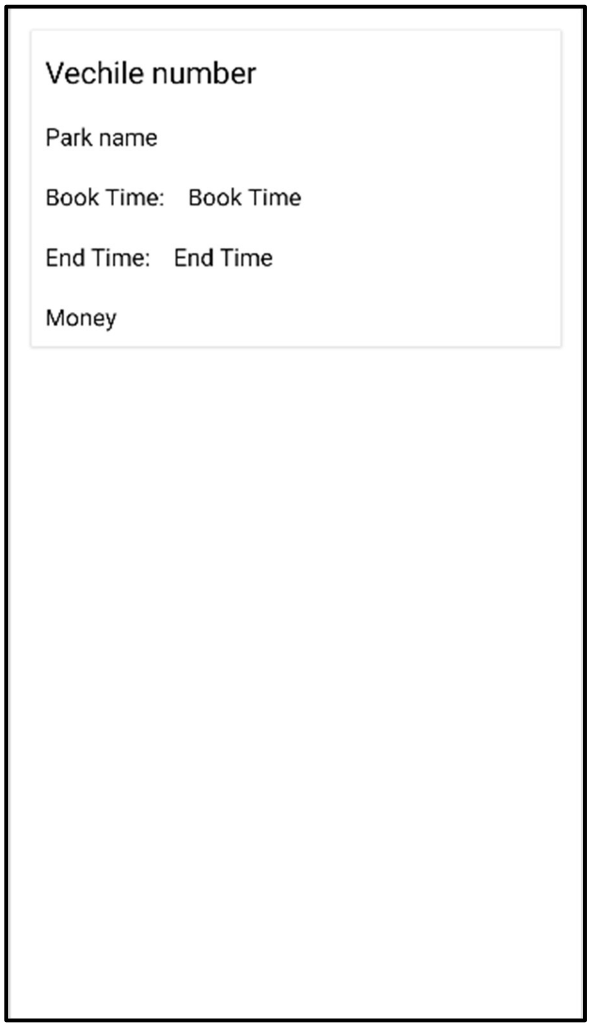


Fig 6.1.9: Items Page

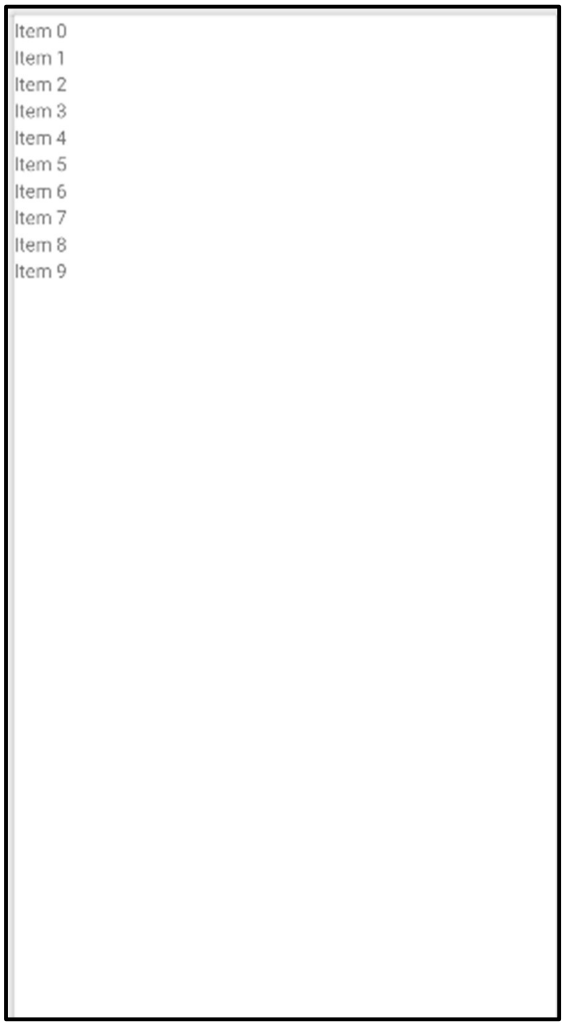


Fig 6.1.10: All Bookings

Description of the above images are mentioned in implementation chapter.

CHAPTER 7

CONCLUSION AND SCOPE

This is the modern age. Many people have vehicles. The vehicle is now a basic need. Every place is in the process of urbanization. There are many corporate offices and shopping centers etc. There are many recreational places where people used to go for refreshments. So, all these places need a parking space where people can park their vehicles safely and easily. Every parking area needs a system that records the detail of vehicles to give the facility. These systems might be computerized or non-computerized. With the help of a computerized system, we can deliver a good service to a customer who wants to park their vehicle on thy organization's premises.

The difficulty of searching for available parking lots has been eliminated by reserving lots via the proposed system. Users can get learn about parking areas for the locations. It saves user time in search of the parking space available in such a long parking area. We will be thankful for your honest review of this software so we can make it even more efficient and update it with new features. In the further development of this Android app plan to make the app more advance by adding more support and additional functions to the system. Also, plan to use a credit card payment gateway.

Spypark is mainly used in big cities where now finding parking space can cause a lot of traffic problems for other vehicles and can take time. So, this version of the computerized program will now help in those fields. Although we have achieved many of our thoughts for this project there are still some which we need to work on. In the future, we would now like to improve financial transactions in the computerized method according to time.

REFERENCES

[1] Porle, R.R. and Saiful, N.N.M., 2021, November. Android-based Booking Application for Smart Parking System. In 2021 IEEE 19th Student Conference on Research and Development (SCOReD) (pp. 290-294). IEEE.

[2] Tu, L., Ma, Z. and Huang, B., 2019, August. Analysis and prediction of differential parking behaviors. In 2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech) (pp. 44-49). IEEE.

[3] Wang, H. and He, W., 2011, April. A reservation-based smart parking system. In 2011 IEEE conference on computer communications workshops (INFOCOM WKSHPS) (pp. 690-695). IEEE.

[4] Raji, C.G., Aboobacker, A.B., Muhammad, A., Jamshidha, K. and Shemeem, J., 2022, December. Android based Integrated Parking System for Real-Time Parking. In 2022 International Conference on Automation, Computing and Renewable Systems (ICACRS) (pp. 304-309). IEEE.

[5] Ning, S., Zhong, J. and Zheng, X., 2021, May. Design of Virtual Intelligent Parking Lot System Based on Signal Request Mechanism. In 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD) (pp. 593-597). IEEE.

[6] Sheelarani, P., Anand, S.P., Shamili, S. and Sruthi, K., 2016, February. Effective car parking reservation system based on internet of things technologies. In 2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave) (pp. 1-4). IEEE.

[7] Hainalkar, G.N. and Vanjale, M.S., 2017, June. Smart parking system with pre & post reservation, billing and traffic app. In 2017 International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 500-505). IEEE.

[8] Karthi, M. and Harris, P., 2016, October. Smart parking with reservation in cloud based environment. In 2016 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM) (pp. 164-167). IEEE.

[9] Fahim, A., Hasan, M. and Chowdhury, M.A., 2021. Smart parking systems: comprehensive review based on various aspects. Heliyon, 7(5), p.e07050.

[10] Moosa, A.A. and Anaee, M.A., 2022. Design and Implementation of an Intelligent Parking System (ISPS) With Laser Security System (LSS). pp.10-20.

[11] Jogada, P. and Warad, V., 2016. Effective car parking reservation system based on internet of things technologies. system, 11, p.12.

[12] Bibi, N., Majid, M.N., Dawood, H. and Guo, P., 2017, March. Automatic parking space detection system. In 2017 2nd international conference on multimedia and image processing (ICMIP) (pp. 11-15). IEEE.

[13] Tandon, R. and Gupta, P.K., 2019. Optimizing smart parking system by using fog computing. In Advances in Computing and Data Sciences: Third International Conference, ICACDS 2019, Ghaziabad, India, April 12–13, 2019, Revised Selected Papers, Part II 3 (pp. 724-737). Springer Singapore.

[14] Vakula, D. and Kolli, Y.K., 2017, December. Low cost smart parking system for smart cities. In 2017 International Conference on Intelligent Sustainable Systems (ICISS) (pp. 280-284). IEEE.

[15] Schebb, N.H., Heus, F., Saenger, T., Karst, U., Irth, H. and Kool, J., 2008. Development of a countergradient parking system for gradient liquid chromatography with online biochemical detection of serine protease inhibitors. Analytical chemistry, 80(17), pp.6764-6772.

[16] Alharbi, A., Halikias, G., Yamin, M. and Abi Sen, A.A., 2021. Web-based framework for smart parking system. International Journal of Information Technology, 13(4), pp.1495-1502.

[17] Agarwal, Y., Ratnani, P., Shah, U. and Jain, P., 2021, May. IoT based smart parking system. In 2021 5th international conference on intelligent computing and control systems (ICICCS) (pp. 464-470). IEEE.

[18] Mufaqih, M.S., Kaburuan, E.R. and Wang, G., 2020. Applying smart parking system with internet of things (IoT) design. In IOP Conference Series: Materials Science and Engineering (Vol. 725, No. 1, p. 012095). IOP Publishing.

[19] Mahmud, S.A., Khan, G.M., Rahman, M. and Zafar, H., 2013. A survey of intelligent car parking system. Journal of applied research and technology, 11(5), pp.714-726.

[20] Saleem, A.A., Siddiqui, H.U.R., Shafique, R., Haider, A. and Ali, M., 2020. A review on smart IOT based parking system. In Recent Advances on Soft Computing and Data Mining: Proceedings of the Fourth International Conference on Soft Computing and Data Mining (SCDM 2020), Melaka, Malaysia, January 22– 23, 2020 (pp. 264-273). Springer International Publishing.

[21] Utpala, K.N., Kumar, N.S., Praneetha, K., Sruthi, D.H. and Varma, K.S.A., 2019. Authenticated IoT based online smart parking system with cloud. Pramana Research Journal, 9(4).

[22] Hanzl, J., 2020. Parking information guidance systems and smart technologies application used in urban areas and multi-storey car parks. Transportation Research Procedia, 44, pp.361-368.

[23] Jameel, F. and Zafar, N.A., 2021, May. Formal Modeling and Automation of E-Payment Smart Parking System. In 2021 International Conference on Digital Futures and Transformative Technologies (ICoDT2) (pp. 1-6). IEEE.

[24] Li, B., Fan, L., Ouyang, Y., Tang, S., Wang, X., Cao, D. and Wang, F.Y., 2022. Online Competition of Trajectory Planning for Automated Parking: Benchmarks, Achievements, Learned Lessons, and Future Perspectives. IEEE Transactions on Intelligent Vehicles.

[25] Migdadi, M.M., Dado, A.R., Safadi, O.A. and Shadid, H., 2018. Online car parking booking system: the case of Jordan. International Journal of Business Information Systems, 28(2), pp.214-245.

APPENDIX – I

FULL NAME	EMAIL ID	PHONE NUMBER	PHOTOGRAPH
Mr. Karthikeyan S	s.karthikeyan@jainuniversity.ac.in	+91 9543277017	
Anusha Sarkar	sarkaranusha9@gmail.com	+91 8617836405	
Rani Kumari	dreamerrani@gmail.com	+91 9008793962	
Uddeshya Gorai	uddeshyagorai.2001@gmail.com	+91 7872955572	

APPENDIX - II

SOURCE CODE

GitHub: https://github.com/UddeshyaGorai12/SPYPARK

https://github.com/rani-rp1/SPYPARK_ADMIN

Note: This application is still in the development phase and shall be proprietary and may be commercialized of the creators wish to do it. The current version of this application has limited functionalities; some new features are yet to be developed, which will be deployed eventually.