SMART PARKING AND RESERVATION SYSTEM

PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT FOR THE REQUIREMENT OF THE AWARD OF DEGREE OF

BACHELOR OF SCIENCE
(INFORMATION TECHNOLOGY)

Submitted by
SIDDHESH BHARAT PAWAR
A-126
YASH RAJESH KUMAR THAKKAR
A-120

Under the esteemed guidance of
Minal Prashant Suryavanshi
Assistant Professor

AND

Himanshu Jani
Assistant Professor

DEPARTMENT OF INFORMATION TECHNOLOGY
Malini Kishor Sanghvi College of Commerce and Economics
(Affiliated to University of Mumbai)
MUMBAI, 400049
MAHARASHTRA
2022-2023

SMART PARKING AND RESERVATION SYSTEM

ABSTRACT

Parking is a problem for almost everyone today so there has to be a solution, which helps getting rid of problems arising due to the lack of a proper parking management system. Although various traditional PGIS (Parking Guidance Information System) exist, they can serve only a few users because it is difficult for such static systems to disseminate information on a wider scale. So, the aim of this study is to provide a dynamic solution by introducing the concept of parking guidance system over the internet and also using one of the latest techniques available today i.e., the QR code for the user’s ease. The system is basically designed for a parking which can further be extended as required. This system enhances the components of existing parking system available in the colleges. This system runs on a mobile phone platform and provides a visual display of parking lots available to the user so that the user can book or reserve a space. The Quick Response QR code is affixed at every parking space. The user can thus select the parking space from the visual display. The user needs to scan the QR Code while parking and unparking the vehicle. The action of the user is then reflected in the database. The android application was thus developed that can incur the parking information which was uploaded on the web map server. This system reduces the time which is involved in searching the parking space thus reducing the fuel consumption, user’s frustration. It reduces vehicle travel time and parking time.

DECLARATION

I hereby declare that the project entitled”**SMART PARKING AND RESERVATION SYSTEM “**.done at **Malini Kishor Sanghvi College**, **Mumbai,** has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfilment of the requirements for the award of degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.

Name and Signature of the Student

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

QR : Quick Response code

PGIS : Parking guidance and Information systems

GPS : Global Positioning System

SMS : Short Message Service

APIS : Advanced Parking Information System

IoT : The Internet of Things

SQL : Structured Query Language

API : Application Programming Interface

ASP : Active Server Pages

OS : Operating Systems

SDK : Software Development Kit

IDE : Integrated Development Environment

ADT : Android Development Tools

NDK : Native Development Kit

APK : Android Application Package

HTML : Hyper Text Markup Language

RDBMS : Relational Data Base Management Systems

XML : Extensible Markup Language

ETL : Extract, Transform and Load

WCF : Windows Communication Foundation

TFVC : Team Foundation Version Control

FDI : Foreign Direct Investment

SDLC : System Development Life Cycle

RAD : Rapid Application Development

ROI : return on investment

DFD : Data Flow Diagram

SSADM : Structured-Systems Analysis and Design Methods

UML : Unified Modeling Language

CHAPTER 1

INTRODUCTION

Use of automobiles is increasing day by day which leads to various parking issues. Vehicular population is shooting out the roof, no amount of space is sufficient to accommodate stationary vehicles. Management of parking has grown to large extent. The main problem is to manage parking in congested areas. One of the congested areas is Shopping malls. However, improving parking on Shopping malls is important. The problem is parking spaces are either insufficient according to the demands of people or these spaces are poorly allocated. Shopping malls have to try almost every possible way to deal with problem of people parking. Parking on shopping mall needs improvement. Users entering the mall are allowed to have a car on shopping mall. With every new freshman entering parking possess a problem in mall. Problems in parking car results in users' inconvenience, which results in frustration.

Parking the car today need parking policies for safety and security reasons. There is always competition for the parking space . A good solution to overcome parking crises would be by increasing the number of parking spaces or else enlarge the parking lots, but this will lead to huge investment. However better management of existing parking spaces will be wise method. The availability of parking spaces should be improved. Another approach for managing parking in mall is by improving the efficiency of the use of existing parking spaces, by informing user about available parking space and guiding him accordingly. Now a day there is growing popularity and affordability of internet –enabled smartphones and because of data available online we can step to solve parking problem. Android smartphone enables user to virtually carry the internet with him.

A. Mobile web Map

It is a service application providing maps. By using map user can find spaces on his phone . Maps act as communication language of distinct information for viewing whether parking space is engaged or not. It will inform user about current status of parking lot .

B. Quick Response(QR) code

It is 2-D barcode which encode numeric and alpha numeric value. QR code encodes binary information into a square matrix of black and white pixels. QR code scanner application is able to decode information encrypted in QR code . QR code is used for allocation and de-allocation of space.

1.1 BACKGROUND

Motor vehicles are a major mode of transportation, which has seen a significant growth over the years. The need for parking spaces is increasing in conjunction with this growth, and becoming a major problem in busy cities. There are many problems associated with this overuse such as pollution, fuel consumption, wasted time because of the looping process, a higher percentage of accidents, and drivers' frustration due to traffic congestion . The open loop strategy is defined as the Blind Search, where drivers keep cruising the area looking for a vacant parking and will stop once they reach a free spot . A study shows that approximately 45% of road traffic is caused by motorists looking for a free parking spot . Another study by Donald Shoup [specified that about 30% of traffic is mainly due to cruising vehicles in congested area such as downtown. Moreover, traditional methods like static or digital parking signs at sites are no longer relevant, because of the significant increase in drivers who are looking for parking. Drivers' eyes are often busy off-road and they lose their focus on what is happening on the road, by searching for parking signs or free parking. Even if they temporarily wait for a spot to be vacant, parking illegally on-road has a direct impact on road traffic . Nowadays, we have advanced technologies that are used in many different fields to solve problems like reservations. Evoking these technologies and merging them with a single travel related system is still in the early stages and needs to be used by drivers in their day-to-day travels. Certain companies are already applying several reservation methodologies and are making huge investments to achieve customer satisfaction and maximize their profits. In the last decade, there have been many online parking reservation systems that were developed to serve people using computer-based web browsers and allow them to book their parking in advance. Unfortunately, these systems were developed to work on personal computers and laptops, before the era of smartphones. With the growth of technology, the idea of finding and reserving parking online can be realized by creating a reservation system that assimilates the concept of a smartphone parking reservation application with parking service providers everywhere. This system will save users time and allow them to know ahead of time when and how to reach their guaranteed parking space, instead of having to travel to their desired destination unsure of where to park their vehicle.

Some major daily parking issues include:

1. Lack of parking spaces, mainly in the urban area
2. Misuse of available parking spaces
3. More time and fuel/gas are used to find open parking spaces
4. Difficulty in finding vehicles at large parking lots
5. Traffic congestion is concentrated on underutilized parking spaces
6. Business parking lots are taken over by passenger parking
7. Incorrect parking
8. Proper management of disabled areas & unused private parking lots
9. The natural impact of excessive fuel consumption in search of a parking space
10. Unclear parking policies.

1.2 OBJECTIVE

Parking support :

The application helps users spot the most convenient place to park their car. The support is twofold: in real- time, it is possible searching for the closest parking location w.r.t. a given target position; for planned trips, positions available for booking can be browsed and reserved in advance. A typical parking procedure starts with a parking search action; then, as the parking place has been reached the actual occupation is carried out by the identification of the spot to place the car on. The system functionalities will be described from the user point of view, showing how the application has to be operated in practice. This approach can be followed in real demonstrations as well. In the First place, the user is required to sign up to the system. Upon authentication through an Android smartphone, the user is then able to search for a convenient parking place within a specific area, to proceed with its occupation, and/or to book a parking spot. The automatic charging of the due fare is integrated in the overall procedure.
At any application access, some previously started procedures may be still ongoing, so the system checks for possible pending reservations or current parking occupancies, which can be dealt with by a simple management console. To initiate a new procedure, the parking locator service can be used to look for a convenient parking place within a specified area. To this aim, the user must provide the locator service with the city name and possibly address, and a “tolerance" radius. For example, , the user is interested in the status of the parking spots in the center of mall, for 2 Km around. When the Search button is tapped, the current parking information is retrieved (through a call to a web service) and displayed on a map view. The user can then decide to head towards one specific place. Possibly, the destination can be reached making use of the satellite navigation system provided by the smartphone. When the user reaches the place and parks the car on the target parking spot, he must perform the next procedure step, i.e., the notification of the start of the occupancy period. This can be simply done by reading the location-specific QR code placed there about , from that moment on, by accessing the management screen, the user is able to track information about his parking slot, such as the effective parking period and the amount to be paid so far. An explicit action is required to signal the end of the parking period, so to pass to the calculation and charging of the due fare: This can be done by a further reading of the QR code. The system also allows for the remote booking of specific, dedicated positions. It is worth noticing that the related billing

policy, as well as the ordinary one for parking, is stated by the parking manager and supported by the backend information system. The main goal of this project is to maximize the occupancy of parking lots and develop a user-friendly mechanism that helps user find and reserve available parking in the shopping mall, in advance. At times of peak parking in the parking lot, the only primitive way is to accurately provide users with available parking spots inside the parking lot.

Mobile Maps and Devices:

Maps in the pre-internet times can be grouped into three categories: view only maps, analytical maps and explorative maps. Though maps still remain the most popular communication language of spatial information the internet and advances in technology has necessitated a creation new kind of maps that will serve the same use as the old maps as well as incorporate dynamism and mobility. Modern maps are no longer used for mere presentation but for interactive and individual exploration of temporal and non-temporal spatial data. The internet has revolutionized the distribution of screen maps with the web-based maps being seen as a metaphor to spatialize the information space and as a collaborative thinking instrument shared by spatially separated users. The realization of wireless internet access has brought web maps back to mobile environments where they are most needed.

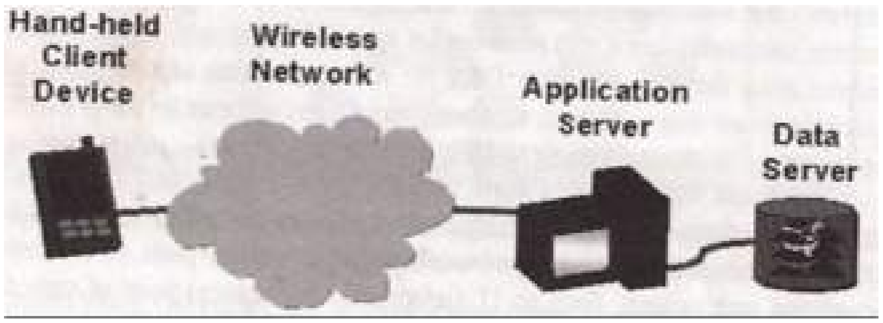


Figure 1: Handheld mobile GIS platform

Unlike web-maps, mobile maps are more personal and provide better platform to relay spatial information especially of a temporal nature. A mobile map is somewhat like a snapshot of an environment around a certain location and time, but with highly selective information and integrated intelligence. Most of our daily activities require us to be in motion and driving is no exception. With parking spaces being engaged on and off, a static or a standalone system, like the case of most PGIS would seem handicapped in relaying parking information. Mobility is unquestionably a fundamental aspect of contemporary life. With mobile maps modern mobile people (drivers) will be better informed of the events from near and far, past, present and future; ensuring they are better prepared for their tasks. There is a wide array of hand-held devices that can be classified into three types based on weight, power, cost and functional capabilities. These are Portable PCs, PDAs and mobile phones. This study focuses on the mobile phone since currently it's the most used type, versatile and most applicable for disseminating parking information to many users at a time. Moreover, a key feature of mobile phones is that they have the ability to determine their location using Global Positioning

Systems, GPS embedded in them. Other characteristics that make mobile phones better suited for mobile GIS include; high mobility, dynamism and ability to operate in real time, supports applications and ability to sense locational information. It's because of the above reasons that such mobile phones have been branded a new name: Smart phones. Smart phones have hit the market with huge sales. Unlike standard cell phones they have additional features that allow the user to do a lot of things on and off the internet. An operating system like Android enables one to log in the internet and install applications suited for his needs. For instance, Quick Response (QR) Codes and Quick Response Readers are features that are gaining popularity across the nations. A smart phone equipped with a QR Reader application is able to decode information encrypted in QR Code and prompt the phone to do necessary action as directed by the QR Code

Figure 2: A QR Code and a smart phone with QR Reader

There are codes that prompt the phone to make a call to a specific number, open the default browser of the phone and visit a particular website, display a message or even prompt the user to send a message to a particular number upon populating the Short Message Service, SMS function of the phone. The SMS-QR Code type is used in this study. The idea is to automate the message sending function of the phone thus reduce time and effort in relaying a SMS.

1.3 PURPOSE , SCOPE AND APPLICABILITY

1.3.1 PURPOSE

The purpose of this project is to make people more convenient to park their vehicle, which in this case is Reservation Based Smart Parking System, the question to be addressed here in this module is, how to give parking slots to the drivers? The project is to mainly answer this particular question addressed by providing an Android application to reserve parking slot as per drivers need.

1.3.2 SCOPE

The goal of smart parking systems is to know where parking is available and to let the driver know as well, making it easier for cars to find their way into parking spots. Another goal is to understand just where parking is in demand and when. For cities, this information can be extremely useful. ”Parking is generally the second or third largest source of revenue for a city. So, there’s a significant financial impact to this.”

1.3.3 APPLICABILITY

Applications which extend the functionality of devices, are written using the Android software development kit (SDK) and often the Java programming language that has complete access to the Android APIs. Java may be combined with C/C++, together with a choice of non-default run times that allow better C++ support the Go programming language is also supported since its version 1.4, which can also be used exclusively although with a restricted set of Android APIs. The SDK includes a comprehensive set of development tools, including a debugger, software libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Initially, Google's supported integrated development environment (IDE) was Eclipse using the Android Development Tools (ADT) plugin in December 2014, Google released Android Studio, based on IntelliJ IDEA, as its primary IDE for Android application development. Other development tools are available, including a native development kit (NDK) for applications or extensions in C or C++, Google App Inventor, a visual environment for novice programmers, and various cross platform mobile web applications frameworks.

1.4 ACHIEVEMENTS

In this proposed design of smart parking and reservation system, this project aims to develop a car park reservation system to address and tackle the above daily problems. This system allows car park operators to manage their carparks and their customers‟ activities, it also includes a mobile-based client for car park searching and making reservation, which would be used to search for and locate the nearby available car park with geometry information as well as reserving for a guaranteed parking space in advance. This project will be focusing on how reservation could be done, and how drivers‟ parking experience could be improved for the above situation through employing and using software system with latest technology. Ultimately, saving in time, fuel and a better parking experience could be achieved by using this system

1.5 ORGANISATION OF REPORT

In Chapter 1, I have described about our project and the background and Objectives in1.1 and 1.2. 1.3 Includes three parts in which I have discussed the purpose scope and applicability from 1.3.1 through 1.3.3.

In Chapter 2, I have discussed about the survey of technology and tools used and its working which is compatible with this project. I have also discussed the various papers that we have referred to for our project. This section includes the title of the papers, along with their description and the pros and cons of those projects. Here, I have also mentioned the ways by which propose to overcome all the disadvantages of the projects that have been described in the paper. This chapter also includes the technological review of this project.

In Chapter 3, the requirement analysis of my project has been discussed. This includes the operating system that we are working on, the hardware, software, front end and the backend requirement of our project to execute successfully. A complete planning and scheduling of this project with problem definition, preliminary product description and conceptual models have been discussed.

Chapter 4, is based on system design. This includes all the design approaches that include the front-end design, component diagram, deployment diagram, E-R diagram and the flow graph of this project. This will also include the basic modules and designs used. This is related to the implementation details of this project. This includes the assumptions that I have taken into consideration while designing our project and also the dependencies, describing the modular of the project. The use case report and the class-diagram report has been explained respectively.

Chapter 5, includes coding details, test cases of the way the project has been tested, detailed information of the types of testing in a software management life cycle including the testing approach used for this particular project with its future modification and

improvement.

Chapter 6, contains the results of the working assembled project after testing it and discussions of end- user usage manual and documentation along with test reports of each component's expected results after testing. The purpose of this documentation is to guide

the users on how to properly install, use, and/or troubleshoot a product and prevent the user directly reaching the customer care.

Chapter 7, has the conclusion, significance of the project, why and how this project system will benefit the future, limitations of the systems and future scope of the system about when can the changes take place adapting the future requirements and how can the system be

compatible with improved features considering the current limitations.

CHAPTER 2

SURVEY OF TECHNOLOGIES

2.1 EXISTING SYSTEM

- **Vision Based Method**

Monitoring detection technology can be divided into two categories. The first estimates the number of remaining vacant spaces for the entire parking lot by counting incoming and outgoing vehicles. The second monitors the status of each individual space and can be used to guide a car to a vacant space. To detect the status of an individual parking space different methods have been utilized, such as ultrasonic sensors placed at each space (thus it requires many sensors), or surveillance cameras placed at a high position.

- **Sensor Based Method**

Another detection technology uses sensors to detect vacant spaces in a parking lot. Different factors play a role in choosing the proper sensor, including size, reliability, adaptation to environmental changes, robustness and cost. Sensor's technologies are categorized as either intrusive or non-intrusive. Intrusive sensors need to be installed directly on the pavement surface, so digging and tunnelling under the road surface are required. Non-intrusive sensors only require fixing on the ceiling or on the ground. Ultrasonic sensors are categorized as non-intrusive sensors. Ultrasonic sensors transmit sound waves between 25 kHz and 50 kHz. They use the reflected energy to analyze and detect the status of a parking space. Ultrasonic waves are emitted from the head of an ultrasonic vehicle detection sensor every 60 milliseconds, and the presence or absence of vehicles is determined by time differences between the emitted and received signals.

- **Two Tier Parking & Automatic Multilevel Car parking System**

Two Tier Car Parking System is ideally suited for people having 2 cars They can use parking space for a single car to park both their cars using the Two-Tier Parking System one above the other. The system consists of a single platform which allows the car that is not used very frequently to be parked on the upper level and the one that is used frequently on the lower level. G offers 2 variants for the Two-Tier Parking System - Hydraulic System and Electro-Mechanical System. Automatic Multilevel Car Parking Systems can be fully automatic or semiautomatic. They can be manned or unmanned systems (i.e., operated manually or using computers). These systems can be installed above or below the ground thereby making optimum use of available space. Another advantage in this case is that human intervention is not required for parking the car.

2.2 PROPOSED SYSTEM

we present the architecture and design of proposed reservation-based smart parking system, which implements a reservation service to reduce the traffic volume caused parking cruise.

System Architecture and Design

Fig. 3 shows three components in the smart parking model, including parking zones, users and the database smart parking system. The management system determines the parking prices and broadcast lives parking availability information to users (also drivers). Upon receiving parking information, the user selects desired parking lot and reserves a space. As soon as user reserves a parking space, System generates a unique QR code and sends it to the user. As a result, the state of parking resources is changed by users parking decisions. The parking lot consists of a group of parking spaces. The state of a parking lot is the number of occupied spaces versus total spaces. Every parking lot has access to the Internet to communicate with the management system and users, and share parking information with other parking lots. In each parking lot, the reservation authority is deployed for authenticating the individual user's identity and reservation request

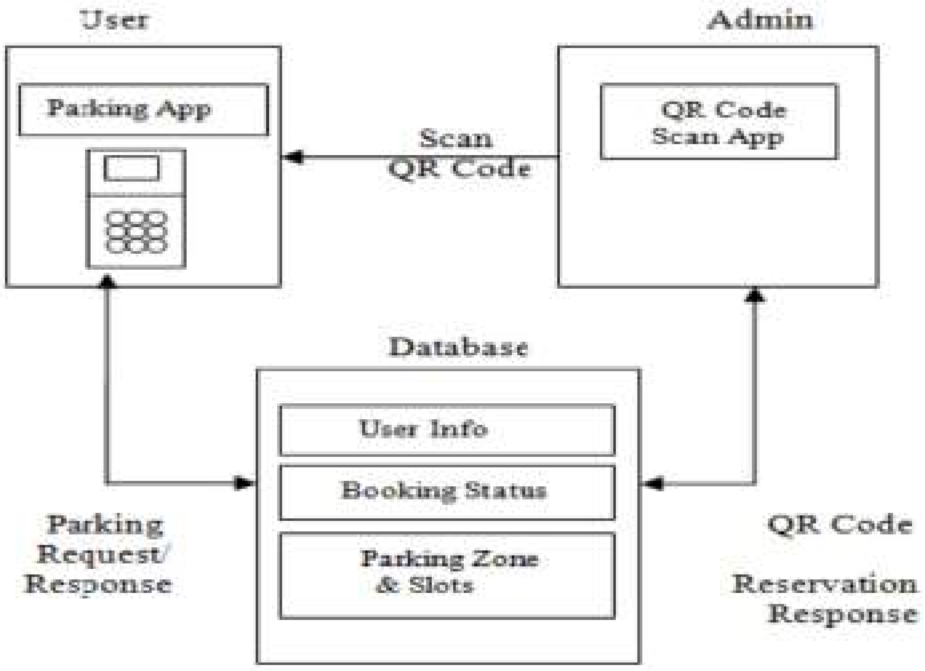


Figure 3: System Architecture

In this case, Reservation authority identifies each user by the unique QR code which has been send by the management system to the user at the time of reservation. Once the reservation order is confirmed, the reservation authority updates reservation information to hold the related space for the user. Upon retrieving the parking information, the system updates the state of the parking lot. Based on the state of parking lots, the system (1) analyses their occupancy status and congestion level, (2) determines the parking prices according to their pricing scheme, (3) broadcasts the prices to all users periodically, and (4) stores the parking information, QR code and prices for further analysis. The system serves as the centralized decision-making body in a planned economy. It makes all pricing decisions regarding the state of parking lots and user demands. This system is a closed-loop system to dynamically adjust parking price, balance the benefits between users, and service providers and reduce traffic searching for parking. By placing the reservation authority on the gate each user has been identified by the QR code, when user reaches the parking spot. Host demands for the QR code and verify the details by scanning the QR code. Since user does not need to communicate with his desired parking lot host to make his reservation, rather he directly scans the QR code by host QR code scanner and verify the details just like a centralized system. Due to this the communication overhead of reservation is highly reduced. Also, since each parking lot manages its own reservation information, it makes the reservation requests from users easily to be synchronized, comparing with reservation synchronization in the system

2.3 HARDWARE REQUIREMENT

The system hardware is organized into three main components, the QR code scanner, the central server and the mobile device, as shown in Fig 4. In the following, we discuss the detailed design and implementation of each component, along with the specification of communication between them.

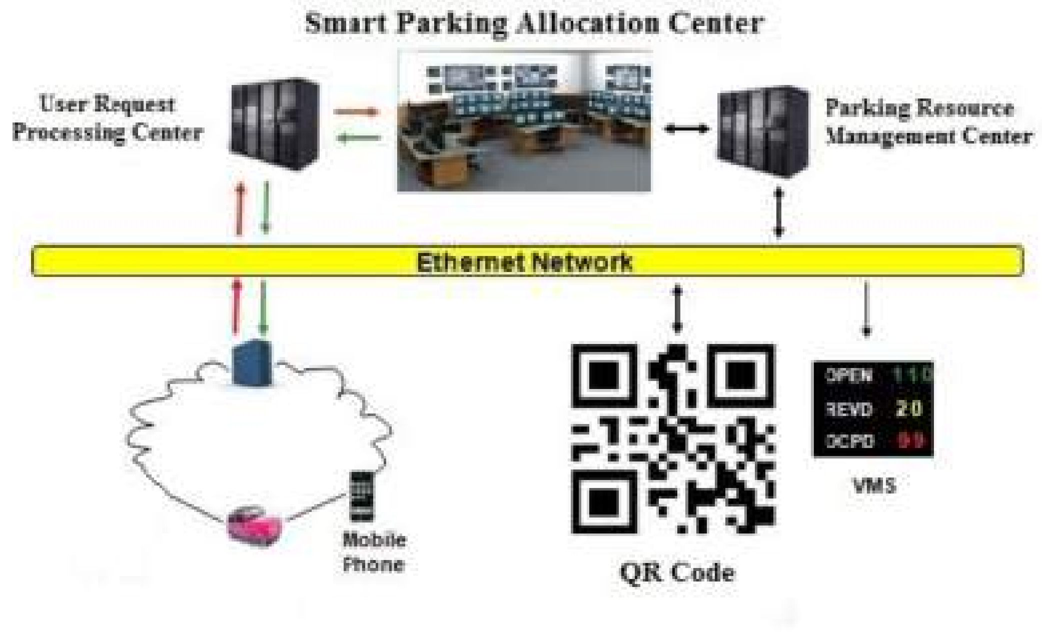

Smart Parking Allocation Center
User Request Processing Center
Parking Resource Management Center
Ethernet Network
OPEN 110
REVD 20
OCPD 99
VMS
Mobile Phone
QR Code

Figure 4: System hardware components

Android Smartphone's and a Central Server. One Android Smartphone is for user which would have parking App and another one is for the admin at the parking lot for Scanning QR code. Both the Phones should have internet connection. The Central Server is connected to both the Smartphone's for performing various SQL operations.

2.4 SOFTWARE REQUIREMENT

Fig. 3 shows the design of software architecture of user API, primarily defining the Android application, which is the central location of the system to user applications and functions also the Host application as the point of control and configuration for the distributed system. Primary software elements are discussed in the following.

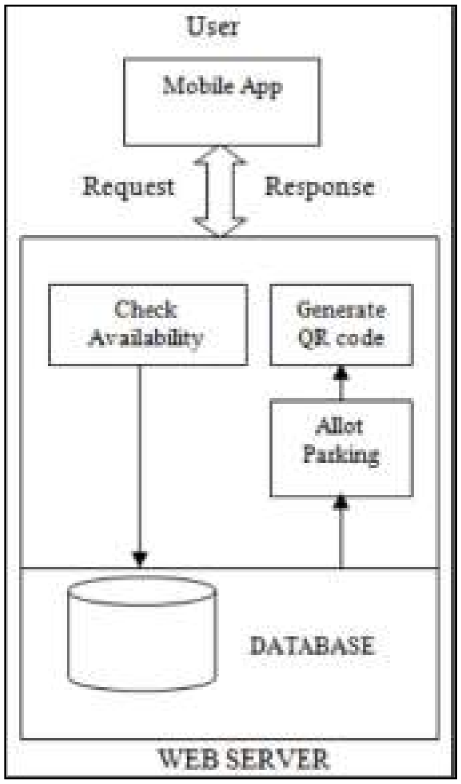


Figure 5: User API

Main System Architecture shows the parking of Smart Parking System based on reservation. The applications are built on Android Platform. Two different apps are used in our System. One is at the user end and another one is for the admin at the parking lot. The Parking app in the user's phone is used to reserve space in desired parking lot. User has to first create an account to be able to use the services provided. Once account is created, user can login with its mobile no as username and password. User can then select appropriate parking lot and check availability. If free spaces are available then user can proceed with space reservation. One user is allowed to reserve only one space. For booking, user has to enter its vehicle's identification number with the start time and end time of reservation. Once Parking space is reserved, a QR code is generated which is used for authentication at the admin end. User is provided with a service that allows user to delay the start time (arrival time) by 15 minutes. If the user is not able to arrive within the extended time, then the reservation is discarded. User is also given a chance to delay the ending time. Prior notifications are sent to users' phone to indicate that reservation time is about to expire (ending time is about to reach). User is then given a chance to extend the ending time. Additional Fares are calculated accordingly for the extended hours. The app at the admin Fig.4 end is used to scan QR code generated in users parking app at the time of reserving space. This makes sure that only users with reservation are allowed to park vehicle. Once QR code in user phone is scanned and is found to be authenticated, database is automatically updated and respective Parking slot status is changed from RESERVED to OCCUPIED. Admin can see all parking slot details. Parking slots would be displayed as graphical boxes coloured as Red, Green and White. Each colour indicates one of the constraints. Green Indicates slot is reserved, White indicates that the slot is free and Red indicates expired slots. Such expired slots have the option of delete which would turn them into free slots.

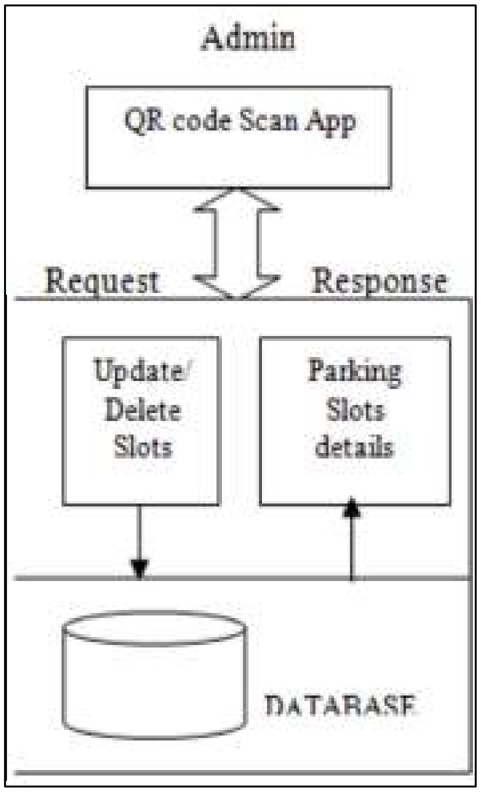


Figure 6: Admin API

2.5 REQUIREMENT ANALYSIS

HARDWARE REQUIREMENTS:

This Application requires user to have Android Smartphone which has active internet connection.

- Desktop / Laptop
- Windows 7 or higher
- Minimum 4 GB RAM
- Minimum 10 GB Memory
- Smartphone
- Android OS Jelly Bean(API level 16) or higher
- Minimum 2 GB RAM
- 10 MB Storage

SOFTWARE REQUIREMENTS:

- Visual Studio
- Android Studio
- SQL Server Management Studio

LANGUAGES USED

- C#
- JAVA
- SQL
- JSON

2.6 JUSTIFICATION OF SELECTION OF TECHNOLOGY

1. ASP.NET

ASP.NET is more than the next version of Active Server Pages (ASP); it is a unified Web development platform that provides the services necessary for developers to build enterprise-class Web applications. While ASP.NET is largely syntax-compatible with ASP, it also provides a new programming model and infrastructure that enables a powerful new class of applications. You can migrate your existing ASP applications by incrementally adding ASP.NET functionality to them. ASP.NET is a compiled .NET Framework -based environment. You can author applications in any .NET Framework compatible language, including Visual Basic and Visual C#. Additionally, the entire .NET Framework platform is available to any ASP.NET application. Developers can easily access the benefits of the .NET Framework, which include a fully managed, protected, and feature-rich application execution environment, simplified development and deployment, and seamless integration with a wide variety of languages.

.NET Framework:

The .NET Framework is Microsoft's Managed Code programming model for building applications on Windows clients, servers, and mobile or embedded devices. Microsoft's .NET Framework is a software technology that is available with several Microsoft Windows operating systems. In the following sections describes , the basics of Microsoft .Net Frame work Technology and its related programming models.

C# is a language for professional programming. C# (pronounced C sharp) is a programming language designed for building a wide range of enterprise applications that run on the .NET Framework. The goal of C# is to provide a simple, safe, modern, object-oriented, high-performance, robust and durable language for .NET development. Also, it enables developers to build solutions for the broadest range of clients, including Web applications, Microsoft Windows Forms-based applications, and thin- and smart-client devices.

2. ANDROID

Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touchscreen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on notebooks, game consoles, digital cameras, and other electronics. Android has the largest installed base of all operating systems (OS) of any kind. Android has been the best-selling OS on tablets since 2013, and on smartphones it is dominant by any metric. Initially developed by Android, Inc., which Google bought in 2005, Android was unveiled in 2007 along with the founding of the Open Handset Alliance – a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. As of July 2013, the Google Play store has had over one million Android applications ("apps") published – including many "business-class apps"] that rival competing mobile platforms – and over 50 billion applications downloaded An April–May 2013 survey of mobile application developers found that 71% of developers create applications for Android, and a 2015 survey found that 40% of full-time professional developers see Android as their priority target platform, which is comparable to Apple's iOS on 37% with both platforms far above others. In September 2015, Android had 1.4 billion monthly active devices. Android's source code is released by Google under open source licenses, although most Android devices ultimately ship with a combination of open source and proprietary software, including proprietary software required for accessing Google services. Android is popular with technology companies that require a ready-made, low-cost and customizable operating system for high-tech devices. Its open nature has encouraged a large community of developers and enthusiasts to use the open-source code as a foundation for community-driven projects, which deliver updates to older devices, add new features for advanced users or bring Android to devices originally shipped with other operating systems. The success of Android has made it a target for patent (and copyright) litigation as part of the so-called "smartphone wars" between technology companies.

3. MICROSOFT SQL SERVER

The SQL Server is a relational database management system from Microsoft. The system is designed and built is to manage and store information. The system supports various business intelligence operations, analytics operations, and transaction processing. The information stored on the server is stored in the relational database. However, since the system is much more than a database, it also comprises of a management system. SQL stands for Structured Query Language, a computer language that manages and administers the server. There are many versions of the SQL server, each subsequent version being an improved model of its predecessor.

Microsoft SQL Server has numerous applications in the business world. The first and most obvious one is the database is used to store and manage information. However, businesses that hold sensitive customer information such as personal details, credit card information, and other confidential information will benefit from increased security. The system also allows the sharing of data files by computers in the same network, a factor that increased reliability. The SQL server is also used to increase the speed with which data is processed, allowing large operations to be executed with ease. With the information stored in the database, businesses will have a reliable backup system.

4. WINDOWS COMMUNICATION FOUNDATION (WCF)

Windows Communication Foundation (WCF) is a framework for building service-oriented applications. Using WCF, you can send data as asynchronous messages from one service endpoint to another. A service endpoint can be part of a continuously available service hosted by IIS, or it can be a service hosted in an application. An endpoint can be a client of a service that requests data from a service endpoint. The messages can be as simple as a single character or word sent as XML, or as complex as a stream of binary data.
While creating such applications was possible prior to the existence of WCF, WCF makes the development of endpoints easier than ever. In summary, WCF is designed to offer a manageable approach to creating Web services and Web service clients.

Features of WCF

WCF includes the following set of features.

- Service Orientation
- Service Metadata
- Data Contracts
- AJAX and REST Support
- Extensibility

5. QR-CODE

QR code (abbreviated from Quick Response Code) is the trademark for a type of matrix barcode (or two-dimensional barcode) first designed in 1994 for the automotive industry in Japan. A barcode is a machine-readable optical label that contains information about the item to which it is attached. A QR code uses four standardized encoding modes (numeric, alphanumeric, byte/binary, and kanji) to efficiently store data; extensions may also be used. A QR code consists of black squares arranged in a square grid on a white background, which can be read by an imaging device such as a camera, and processed using Reed-Solomon error correction until the image can be appropriately interpreted. The required data is then extracted from patterns that are present in both horizontal and vertical components of the image. It is now widely used around the world to get to websites quicker, and it can also be used for advertisements.

The Quick Response (QR code) system became popular outside the automotive industry due to its fast readability and greater storage capacity compared to standard UPC barcodes. Applications include product tracking, item identification, time tracking, document management, and general marketing.

6. VISUAL STUDIO

The Visual Studio integrated development environment is a creative launching pad that you can use to edit, debug, and build code, and then publish an app. An integrated development environment (IDE) is a feature-rich program that can be used for many aspects of software development. Over and above the standard editor and debugger that most IDEs provide, Visual Studio includes compilers, code completion tools, graphical designers, and many more features to ease the software development process.

Visual Studio with an open project and several key tool windows you'll likely uses:

- Solution Explorer (top right) lets you view, navigate, and manage your code files. Solution Explorer can help organize your code by grouping the files into solutions and projects.
- The editor window (centre), where you'll likely spend a majority of your time, displays file contents. This is where you can edit code or design a user interface such as a window with buttons and text boxes.
- Team Explorer (bottom right) lets you track work items and share code with others using version control technologies such as Git and Team Foundation Version Control (TFVC).

Popular productivity features

- Quick Actions
- Code Clean-up
- Refactoring
- IntelliSense
- Search box
- Live Share
- Call Hierarchy
- CodeLens
- Go To Definition

CHAPTER 3

REQUIREMENT AND ANALYSIS

Requirement Analysis, also known as Requirement Engineering, is the process of defining user expectations for a new software being built or modified. In software engineering, it is sometimes referred to loosely by names such as requirements gathering or requirements capturing. Requirements analysis encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product or project, taking account of the possibly conflicting requirements of the various stakeholders, analysing, documenting, validating and managing software or system requirements. Here are the objectives for performing requirement analysis in the early stage of a software project:

- **From What to How**: Software engineering task bridging the gap between system requirements engineering and software design.
- **3 Orthogonal Views**: Provides software designer with a model of:
    - system information (static view)
    - function (functional view)
    - behaviour (dynamic view)

Typically, you carry out the requirements analysis process before you begin project planning. Additionally, in project management, you conduct requirements analysis when:

- Calculating development costs
- Setting project priorities
- Creating a work breakdown structure
- Including project specialists in an ongoing project

The return on investment for a good quality requirements gathering and analysis almost always outweighs the cost. Giving due time and effort into the process means you can deliver a superior product with much fewer roadblocks taking up your time. Some of the benefits of a good requirements analysis include:

- Fewer defects in the finished product
- Faster delivery
- Reducing miscommunication and rework
- Fostering a more collaborative work environment for your team
- Discovering new opportunities for growth and innovation
- Higher customer satisfaction
- Higher team member satisfaction

3.1 PROBLEM DEFINITION

The automotive market which started opening for Foreign Direct Investment (FDI) in 1991, brought many global players to build automobile manufacturing capacities in India. Combined with the increased buying capacity of the middle class of India, people bought multiple automobiles, and changed them frequently, as automobiles historically have been treated as a status symbol in India rather than a utility item. This led to an explosion in the number of automobiles on the roads.

Unfortunately, while the open market also brought top-end, world-class vehicles from Rolls Royce, Audi, BMW and Mercedes, the crudely assembled e-rickshaws on the lowest end of the spectrum, also plied on the same roads. And if this was not enough, there were confusing guidelines for disposal of older vehicles or their movement including control of the movement of many non-standard modes of transport on the roads, a wide range of mobility vehicles started crowding the Indian roads, creating a chaotic mobility network in India.

Although a lot of work has been done on the development of expressways, highways, and other related infrastructure, roads within the city and their connectivity with highways could not progress much. While many other systems are being regularly identified and acted upon to improve the Indian mobility network, the parking system never got its due and remained the most neglected system in the mobility chain of India.

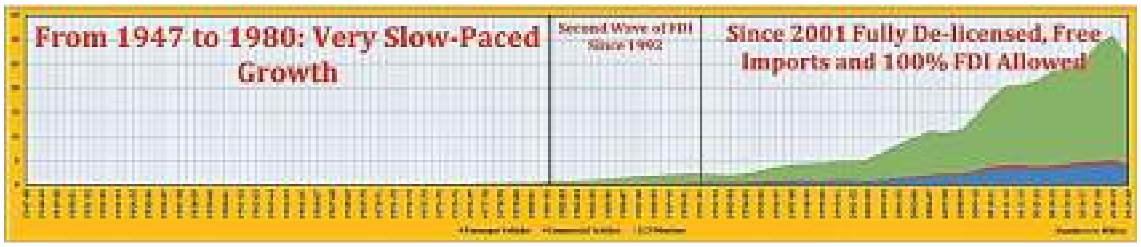


Figure 7: Exponential Rise of Number of Automobiles with liberalisation

With outdated, unplanned manual parking systems lacking any kind of discipline, the general tendency of people in India has been to park their cars anywhere they want to. This approach which is due to the unavailability of a proper parking place, causes traffic jams, unsafe parking, damage to automobiles and even accidents. As a result, chaotic situation is common on almost all Indian roads, markets, and in public places.

The parking system remains one of the lowest priority areas and is one of the most problematic in the mobility chain. The number of vehicles continues to outnumber the existing parking spaces. These haphazardly grown horizontally spread parking areas have been one of the most serious social issues leading to congestion and/or choking of roads, eating of pedestrian pathways/ green belts/ water bodies, contaminating the land area, etc. The single-level unplanned parking, which are often created on a part of the public road, also leads to jumbling of different modes of transportation confusing traffic movement, and, as such, neither lane nor sane driving can be followed nor enforced. Due to such chaos, a lot of valuable time is wasted. For places like shopping malls and amusement parks, it causes economical loss and a lot of people are unwilling to visit these places as they spend more time in parking and in retrieving their vehicles.

The three main problems that the increasing number of vehicles and the decreasing efficiency of modern busy parking lots are:

1. **Valuable time wasted from inconvenient and inefficient parking lots**

- On average, 3.5 - 12 minutes spent waiting for a spot in urban parking lots

2. **More fuel consumed while idling or driving around parking lots, leading to more CO2 emissions being produced**

- Average distance travelled looking for a spot = 1.2km
- Average CO2 produced per car per day = 14 kg CO2
- 14kg for 1000 cars per day and 5110kg per year just for 1000 cars!

3. **Potential accidents caused by abundance of moving vehicles in disorganized parking lots**

- 413 accidents occurred in public parking lots in India last year
- There were 788 parked car collisions, 5 being fatal
- 2/3 of traffic accidents in parking lots involve only 1 moving vehicle
- Parking Structure Issue!
- 1/3 of these accidents involved 2 moving vehicles
- Parking System Issue!

3.2 MODULE DIVISION

- **ADMIN MODULE (ANDROID APP):**
    1. The admin can login into app.
    2. The admin can scan QRcode.
    3. The admin can accept the payment of users.
    4. The admin can manage users.

- **USER MODULE (ANDROID APP):**
    1. User can login into the app.
    2. Search area for parking.
    3. Book parking space using QR code.

- **ADMIN MODULE (WEB APPLICATION):**
    1. The admin can login into the website.
    2. Manage parking lots.
    3. View users.
    4. Manage payments

3.3 SYSTEM DEVELOPMENT LIFE CYCLE

The System Development Life Cycle is the process of developing information systems through investigation, analysis, design, implementation, and maintenance. The System Development Life Cycle (SDLC) is also known as Information Systems Development or Application Development.

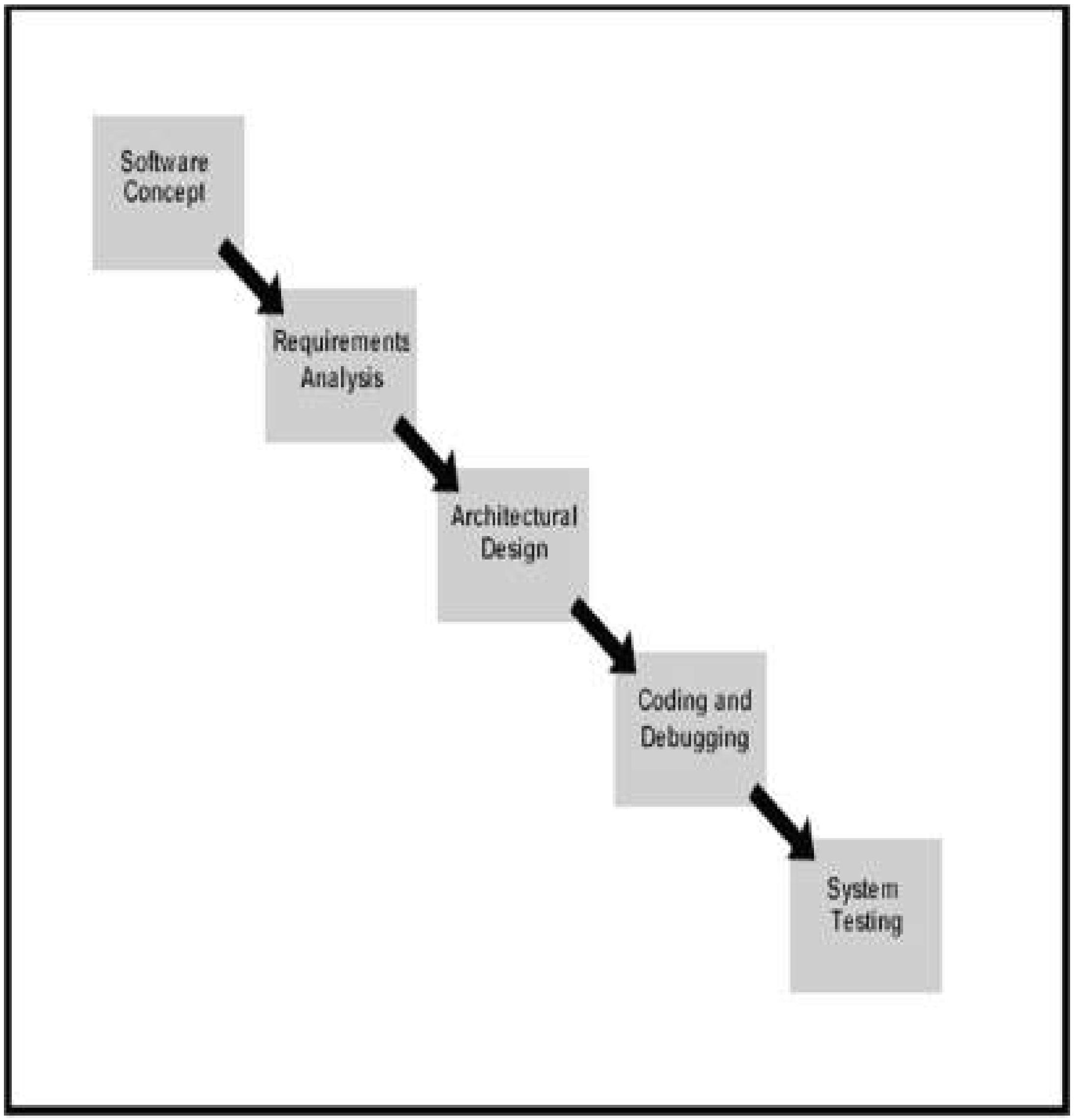


Figure 8: System Development Life Cycle

➔ Steps involved in the System Development Life Cycle :

Below are the steps involved in the System Development Life Cycle. Each phase within the overall cycle may be made up of several steps.

Step 1: Software Concept

The first step is to identify a need for the new system. This will include determining whether a business problem or opportunity exists, conducting a feasibility study to determine if the proposed solution is cost effective, and developing a project plan.

This process may involve end users who come up with an idea for improving their work. Ideally, the process occurs in tandem with a review of the organization's strategic plan to ensure that IT is being used to help the organization achieve its strategic objectives. Management may need to approve concept ideas before any money is budgeted for its development.

Step 2: Requirements Analysis

Requirements analysis is the process of analysing the information needs of the end users, the organizational environment, and any system presently being used, developing the functional requirements of a system that can meet the needs of the users. Also, the requirements should be recorded in a document, email, user interface storyboard, executable prototype, or some other form. The requirements documentation should be referred to throughout the rest of the system development process to ensure the developing project aligns with user needs and requirements.

Professionals must involve end users in this process to ensure that the new system will function adequately and meets their needs and expectations.

Step 3: Architectural Design

After the requirements have been determined, the necessary specifications for the hardware, software, people, and data resources, and the information products that will satisfy the functional requirements of the proposed system can be determined. The design will serve as a blueprint for the system and helps detect problems before these errors or problems are built into the final system. Professionals create the system design, but must review their work with the users to ensure the design meets users' needs.

Step 4: Coding and Debugging

Coding and debugging are the act of creating the final system. This step is done by software developer.

Step 5: System Testing

The system must be tested to evaluate its actual functionality in relation to expected or intended functionality. Some other issues to consider during this stage would be converting old data into the new system and training employees to use the new system. End users will be key in determining whether the developed system meets the intended requirements, and the extent to which the system is actually used.

Step 6: Maintenance

Inevitably the system will need maintenance. Software will definitely undergo change once it is delivered to the customer. There are many reasons for the change. Change could happen because of some unexpected input values into the system. In addition, the changes in the system could directly affect the software operations. The software should be developed to accommodate changes that could happen during the post implementation period.

There are various software process models like:-

- Prototyping Model
- RAD Model
- The Spiral Model
- The Waterfall Model
- The Iterative Model

Of all these process models we’ve used the Iterative model(The Linear Sequential Model) for the development of our project.

3.3.1 SPIRAL MODEL

Spiral model is one of the most important Software Development Life Cycle models, which provides support for **Risk Handling**. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a **Phase of the software development process.** The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using the spiral model.

The Radius of the spiral at any point represents the expenses(cost) of the project so far, and the angular dimension represents the progress made so far in the current phase. .

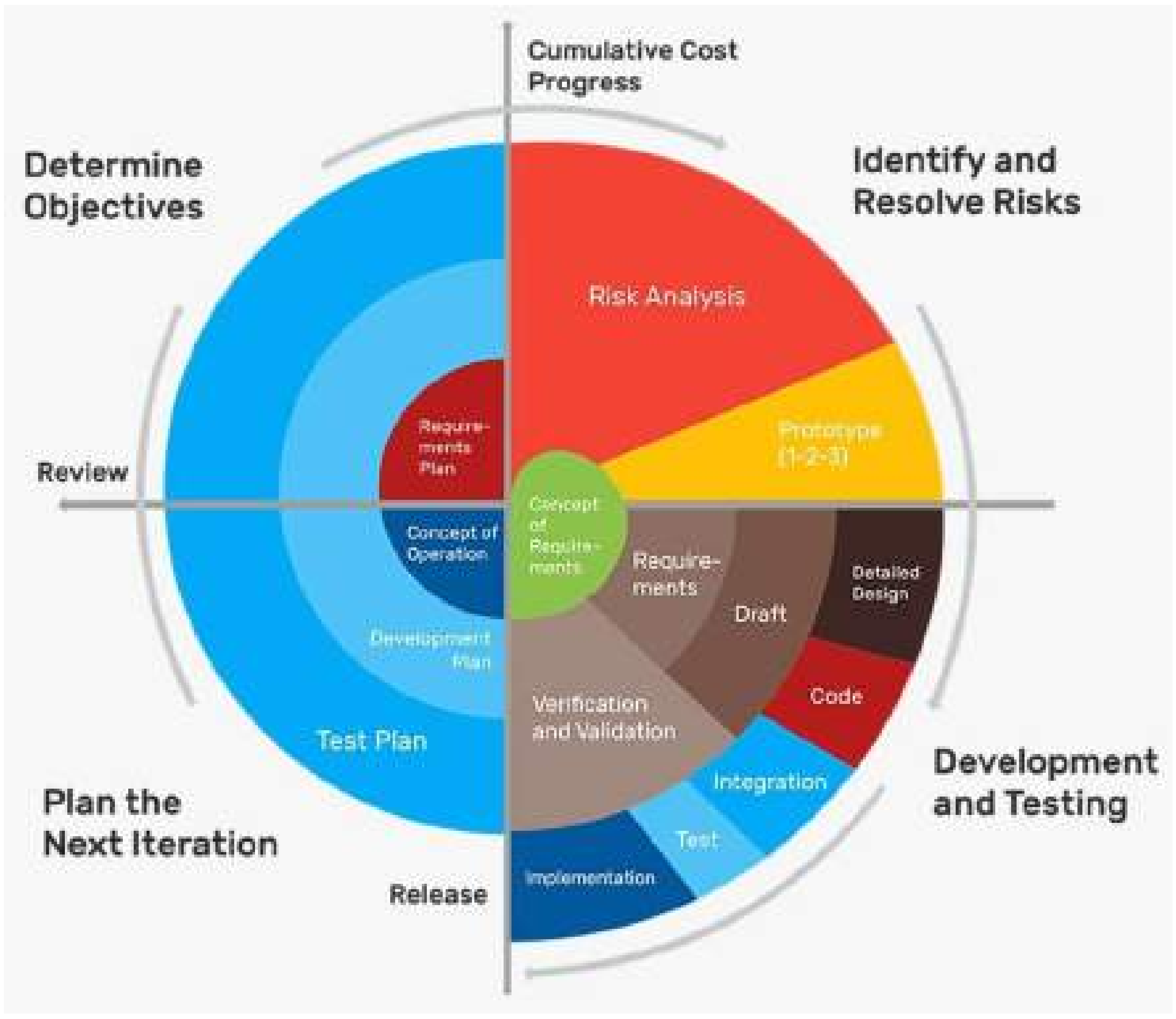


Figure 9: Spiral Model

Each phase of the Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are discussed below-

1. **Objectives determination and identify alternative solutions:** Requirements are gathered from the customers and the objectives are identified, elaborated, and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.
2. **Identify and resolve Risks:** During the second quadrant, all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the Prototype is built for the best possible solution.
3. **Develop next version of the Product:** During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.
4. **Review and plan for the next Phase:** In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next phase is started.

Risk Handling in Spiral Model

A risk is any adverse situation that might affect the successful completion of a software project. The most important feature of the spiral model is handling these unknown risks after the project has started. Such risk resolutions are easier done by developing a prototype. The spiral model supports coping up with risks by providing the scope to build a prototype at every phase of the software development.

The Prototyping Model also supports risk handling, but the risks must be identified completely before the start of the development work of the project. But in real life project risk may occur after the development work starts, in that case, we cannot use the Prototyping Model. In each phase of the Spiral Model, the features of the product dated and analysed, and the risks at that point in time are identified and are resolved through prototyping. Thus, this model is much more flexible compared to other SDLC models.

Why Spiral Model is called Meta Model?

The Spiral model is called a Meta-Model because it subsumes all the other SDLC models. For example, a single loop spiral actually represents the Iterative Waterfall Model. The spiral model incorporates the stepwise approach of the Classical Waterfall Model. The spiral model uses the approach of the Prototyping Model by building a prototype at the start of each phase as a risk-handling technique. Also, the spiral model can be considered as supporting the Evolutionary model – the iterations along the spiral can be considered as evolutionary levels through which the complete system is built.

Advantages of Spiral Model:

Below are some advantages of the Spiral Model.

- **Risk Handling:** The projects with many unknown risks that occur as the development proceeds, in that case, Spiral Model is the best development model to follow due to the risk analysis and risk handling at every phase.
- **Good for large projects:** It is recommended to use the Spiral Model in large and complex projects.
- **Flexibility in Requirements**: Change requests in the Requirements at later phase can be incorporated accurately by using this model.
- **Customer Satisfaction**: Customer can see the development of the product at the early phase of the software development and thus, they habituated with the system by using it before completion of the total product.

Disadvantages of Spiral Model:

Below are some main disadvantages of the spiral model.

- **Complex:** The Spiral Model is much more complex than other SDLC models.
- **Expensive:** Spiral Model is not suitable for small projects as it is expensive.
- **Too much dependability on Risk Analysis:** The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced experts, it is going to be a failure to develop a project using this model.
- **Difficulty in time management:** As the number of phases is unknown at the start of the project, so time estimation is very difficult.

3.4 SYSTEM ANALYSIS

3.4.1 FEASIBILITY STUDY

A feasibility study is an analysis that takes all of a project‘s relevant factors into account—including economic, technical, legal, and scheduling considerations—to ascertain the likelihood of completing the project successfully. Project managers use feasibility studies to discern the pros and cons of undertaking a project before they invest a lot of time and money into it. Feasibility studies also can provide a company‘s management with crucial information that could prevent the company from entering blindly into risky businesses.

A feasibility study is simply an assessment of the practicality of a proposed plan or project. As the name implies, these studies ask: ―Is this project feasible? Do we have the people, tools, technology, and resources necessary for this project to succeed?‖ Also, ―Will the project get us the return on investment (ROI) that we need and expect?‖

The goal of a feasibility study is to thoroughly understand all aspects of a project, concept, or plan; become aware of any potential problems that could occur while implementing the project; and determine if, after considering all significant factors, the project is viable—that is, worth undertaking.

Feasibility studies are important to business development. They can allow a business to address where and how it will operate; identify potential obstacles that may impede its operations, and recognize the amount of funding it will need to get the business up and running. Feasibility studies also can lead to marketing strategies that could help convince investors or banks that investing in a particular project or business is a wise choice.

Technical Feasibility

In technical feasibility the following issues are taken into consideration.

- Whether the required technology is available or not.
- Whether the required resources are available.
    - Manpower- programmers, testers & debuggers.
    - Software and hardware

Once the technical feasibility is established, it is important to consider the monetary factors also. Since it might happen that developing a particular system may be technically possible but it may require huge investments and benefits may be less. For evaluating this, economic feasibility of the proposed system is carried out.

Operational Feasibility

Operational feasibility is mainly concerned with issues like whether the system will be used if it is developed and implemented. Whether there will be resistance from users that will affect the possible application benefits? The essential questions that help in testing the operational feasibility of a system are following.

- Does management support the project?
- Are the users not happy with current business practices? Will it reduce the time (operation) considerably? If yes, then they will welcome the change and the new system.
- Have the users been involved in the planning and development of the project? Early involvement reduces the probability of resistance towards the new system.
- Will the proposed system really benefit the organization? Does the overall response increase? Will accessibility of information be lost? Will the system effect the customers in considerable way?

3.5 SYSTEM PLANNING AND SCHEDULING

3.5.1 PERT CHART

A PERT chart is a visual project management tool that's useful for mapping out project tasks and planning the overall project schedule. Many people confuse PERT with PERT chart, so the best way to provide a comprehensive PERT chart definition is to start with clarifying both of these terms.

PERT stands for Program Evaluation Review Technique. PERT is the actual technique that is used to create a PERT chart. Meanwhile, a PERT chart is the visual diagram that results from using PERT. Think of PERT as the process and the PERT chart as the outcome. PERT charts allow project managers to see essential scheduling details such as task dependencies, task duration estimates, and the minimum amount of time a project can be completed within. But, they're not the most user-friendly or well-understood tool.

DIFFERENCE BETWEEN PERT CHART AND GANTT CHART

Gantt charts are a popular tool for visualizing project schedules. They're a type of bar chart that also shows task dependencies and project timelines.

Gantt charts tend to be less technical and easier for viewers to understand as they don't include the same level of detail as PERT charts. For this reason, PERT charts tend to be better for initially estimating project timelines, while Gantt charts are often preferred for team project time tracking and keeping stakeholders up-to-date during the project.

ADVANTAGES AND DISADVANTAGES OF PERT CHART

PERT chart advantages include:

- Highlighting task dependencies and your critical path.
- More accurate estimates than when relying on single task duration estimates.
- Easy to see how long non-critical tasks can be delayed without impacting the project end date.

PERT chart disadvantages include:

- Durations are still based on estimates. If you did a poor job coming up with your numbers, this method won't be any more accurate than the critical path method.
- PERT charts can be time-consuming and difficult to create and update.
- They can be confusing for stakeholders to interpret.
- PERT assumes one task must end before the next starts. Other types of dependency relationships won't work well with this technique.

PERT TABLE

Activity	Duration (Days)	Immediate Predecessor Activities
Requirement Analysis (A)	20	
Design (B)	25	A
Coding and Debugging I	55	B
Testing and Implementation (D)	10	B and C
Documentation I	10	D

Table 1:Pert Table

PERT Chart

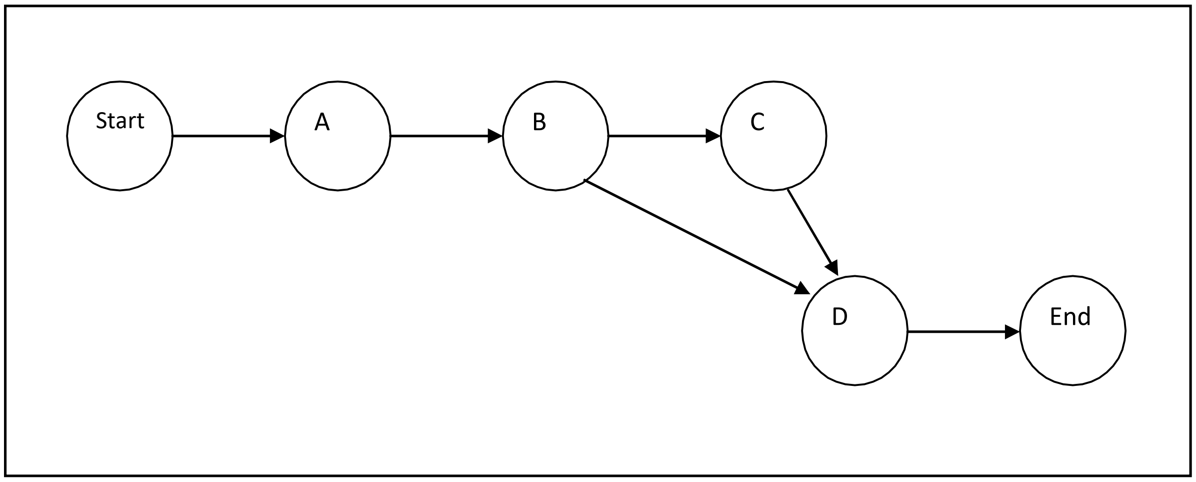


Figure 10: Pert Chart

3.5.2 GANTT CHART

A Gantt chart is a horizontal bar chart used to display the start date and duration of each task that makes up a project. It is one of the most popular project management tools, allowing project managers to view the progress of a project at a glance. Although the Gantt chart is named after Henry Gantt, an American engineer and project management consultant, he was not the first one to devise it. It was devised first by Karol Adamiecki, a Polish engineer, in the 1890s. He created it for his steelworks unit but Henry Gantt customized it for his clients. Today, Gantt charts are used most popularly for project scheduling and control.

Gantt charts make it easy for project managers to identify the critical path to project completion and ensure that there is no delay in those tasks. Project managers should use Gantt charts for project planning and scheduling, allocating resources, tracking the progress of each task at all times and ensuring the smooth and timely execution of critical tasks
Before you create a Gantt chart, gather the following information:

- List of tasks
- Start and end dates for each task
- Task dependencies
- Task owners
- Team members allocated to each task

When you sit down with your team, the first thing to create is a list of tasks that will make up the project. After that, you must estimate the duration, resources required, dependencies, etc. for each task. This will help you in allocating the resources properly. However, you must be prepared to replan and reallocate your resources as the project progresses.

Remember that the basic information needed to create a Gantt chart is a list of tasks with their start and end dates. Depending upon the complexity allowed by your Gantt chart tool, you can add in more details like task dependencies, critical tasks, resource allocated, etc.

3.5.2.1 GANTT CHART

- **GANTT CHART**

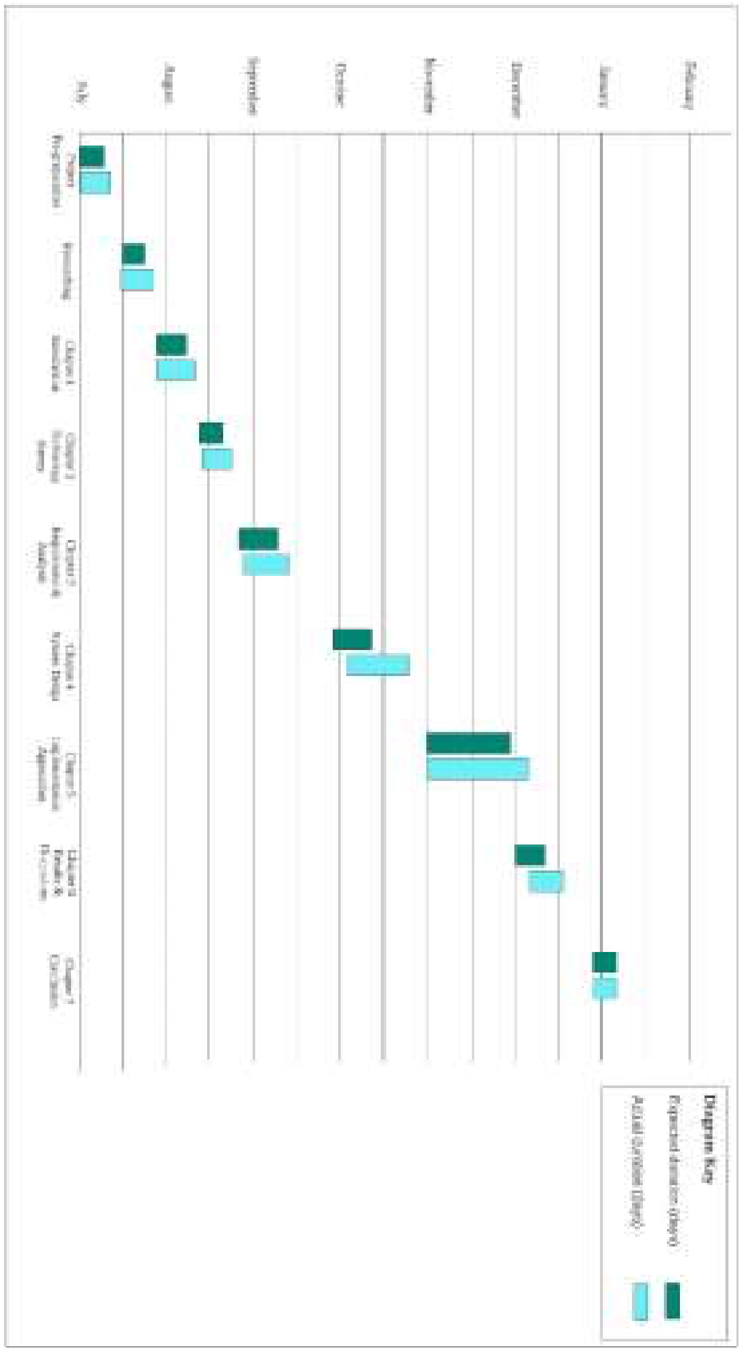


Figure 11: Gantt Chart

3.6 OBJECT ORIENTED ANALYSIS AND DESIGN DIAGRAMS

3.6.1 DFD DIAGRAM

DFD is known to be an abbreviation for a “Data Flow Diagram”. Being a graphical representation of the data flow through an information system, any DFD can be used to model the process aspects of such a system. Being often used as a preliminary step for creating an overview of the system without going into too many details, Data Flow Diagrams may be also used for visualizing some kind of data processing within the study of structured design. A Data Flow Diagram can also be used to display what information can be entered and output from the system. It may represent the way some data can advance through the system, including the description of the place where such data can be stored. Without showing any information about process timing, Data Flow Diagrams are also known to be called the “Bubble charts”.

DFD is a design tool that used in a top-down approach to System Design, which is known to be next "exploded" to create DFD level 1 showing some details of the system that is modelled. By identifying the internal data stores that must be present for the system to do its job properly, and also showing the data flow between different parts of the system, Data Flow Diagrams are one of the well-known significant prospects for structured-systems analysis and design methods called “SSADM”. Having a Data Flow Diagram, users are able to visualize how a system can work, what it can do, and how it can be implemented. Data Flow Diagrams can also be used to provide end users with some physical idea of whether their data affects the structure of the system as a whole. Thus, in order to describe the way some system is being developed, anyone can do it through a data flow diagram model that can be created in the Concept Draw DIAGRAM diagramming and drawing software.

The use of Data Flow Diagrams (DFD) solution as an extension to the Concept Draw DIAGRAM tool can simplify its users' work by creating any necessary Data Flow Diagram, which can help simulate multiple data flows as well as illustrate the functional requirements of the system or business process. Using the pre-made templates and samples of the DFDs, and the stencil libraries full of DFD-related design elements, any Concept Draw DIAGRAM user might find the Data Flow Diagrams (DFD) solution a useful tool for making the needed DFD drawings.

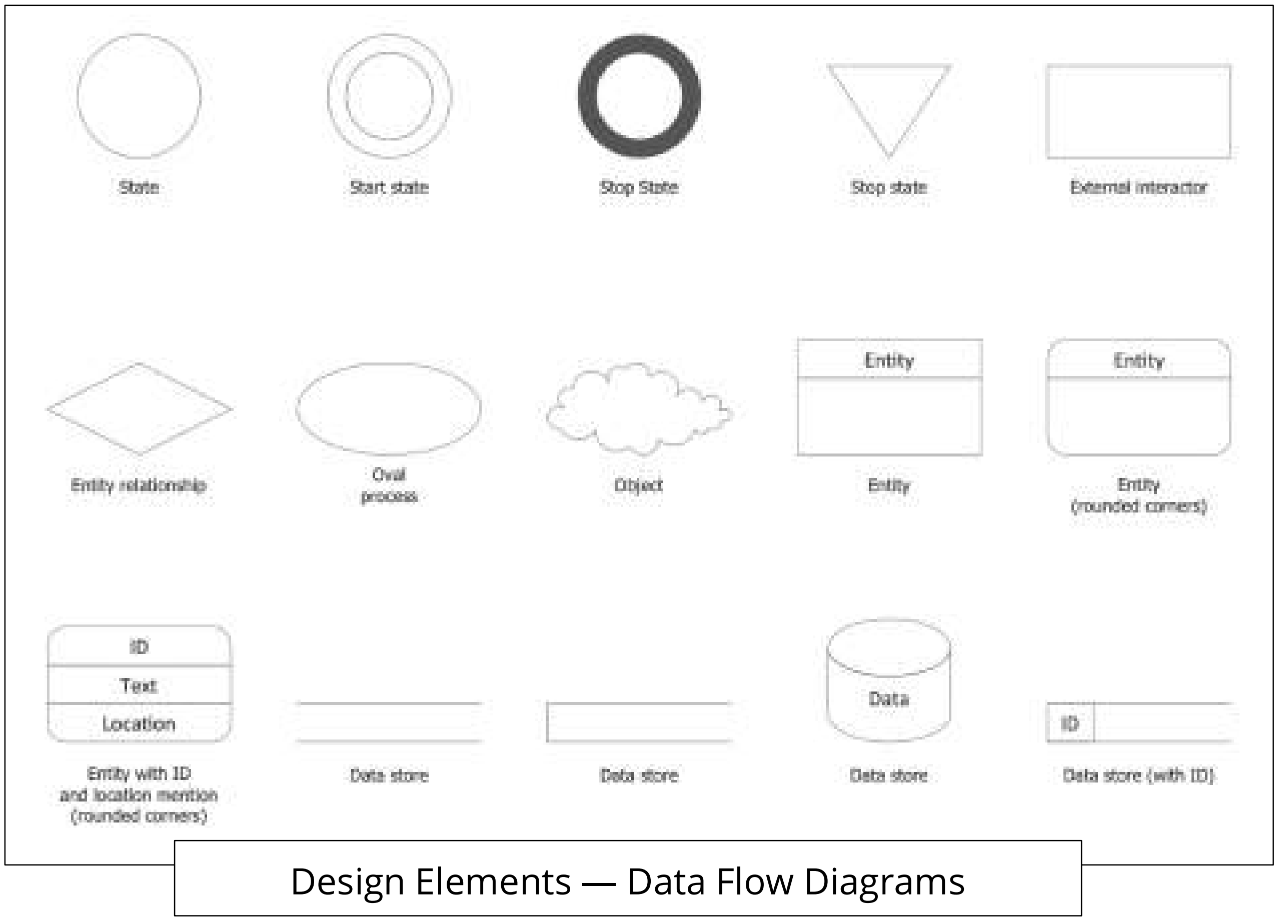


Figure 12: Data Flow Diagram

Parking System Data flow diagram is often used as a preliminary step to create an overview of the Parking without going into great detail, which can later be elaborated it normally consists of overall application dataflow and processes of the Parking process. It contains all of the user flow and their Ent ties such all the flow of Parking Slots, Vehicles, Parking Fees, Duration, Types, Customers, Login. All of the below diagrams have been used for the visualization of data processing and structured design of the Parking process and working flow.

DFD DIAGRAM (DATA FLOW DIAGRAM OR CONTEXT DIAGRAM)

- **DFD LEVEL 0**

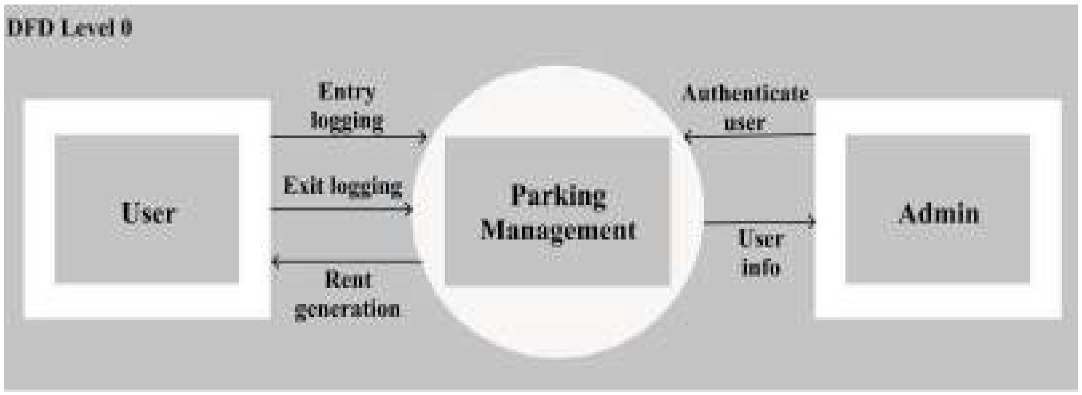


Figure 13: DFD Level 0

- **DFD LEVEL 1**

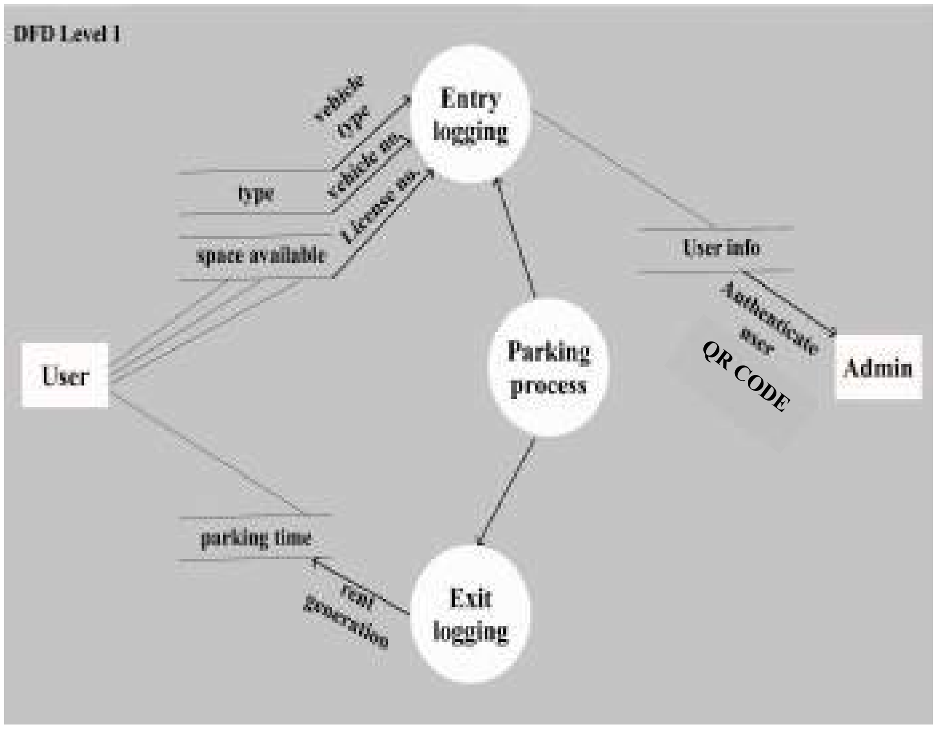


Figure 14: DFD Level 1

3.6.2 ER DIAGRAM (ENTITY RELATIONSHIP DIAGRAM)

ER Model stands for Entity Relationship Model is a high-level conceptual data model diagram. ER model helps to systematically analyze data requirements to produce a well-designed database. The ER Model represents real-world entities and the relationships between them. Creating an ER Model in DBMS is considered as a best practice before implementing your database.ER Modeling helps you to analyze data requirements systematically to produce a well-designed database. So, it is considered a best practice to complete ER modeling before implementing your database.

ER diagrams are visual tools that are helpful to represent the ER model. Peter Chen proposed ER Diagram in 1971 to create a uniform convention that can be used for relational databases and networks. He aimed to use an ER model as a conceptual Modelling approach.

Here, are prime reasons for using the ER Diagram

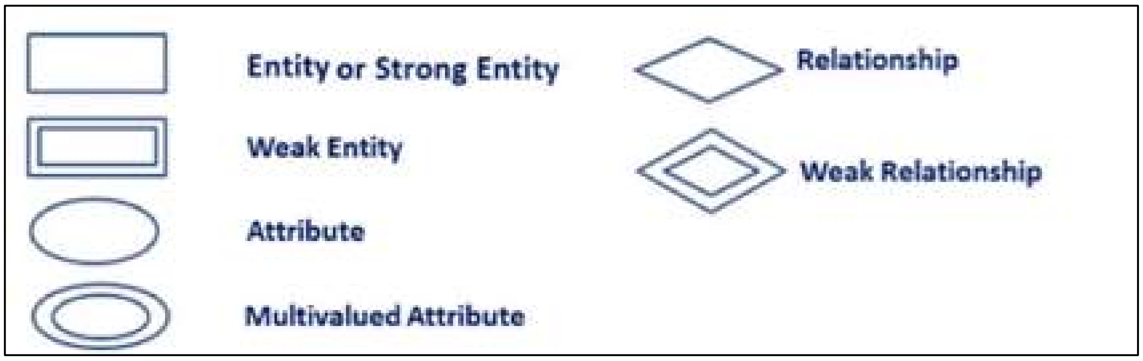
- Helps you to define terms related to entity relationship Modelling
- Provide a preview of how all your tables should connect, what fields are going to be on each table
- Helps to describe entities, attributes, relationships
- ER diagrams are translatable into relational tables which allows you to build databases quickly
- ER diagrams can be used by database designers as a blueprint for implementing data in specific software applications
- The database designer gains a better understanding of the information to be contained in the database with the help of ER diagram
- ER Diagram allows you to communicate with the logical structure of the database to users

3.6.2.1 ER DIAGRAM SYMBOLS AND NOTATION

Entity Relationship Diagram Symbols & Notations mainly contains three basic symbols which are rectangle, oval and diamond to represent relationships between elements, entities and attributes. There are some sub-elements which are based on main elements in ERD Diagram. ER Diagram is a visual representation of data that describes how data is related to each other using different ERD Symbols and Notations.

Following are the main components and its symbols in ER Diagrams:

- **Rectangles:** This Entity Relationship Diagram symbol represents entity types
- **Ellipses :** Symbol represent attributes
- **Diamonds:** This symbol represents relationship types
- **Lines:** It links attributes to entity types and entity types with other relationship types
- **Primary key:** attributes are underlined
- **Double Ellipses:** Represent multi-valued attributes



ER DIAGRAM SYMBOLS

`Figure 15:ER Diagram Symbol

3.6.2.2 COMPONENTS OF THE ER DIAGRAM

This model is based on three basic concepts:

- Entities
- Attributes
- Relationships

ER Diagram Examples

For example, in a University database, we might have entities for Students, Courses, and Lecturers. Students' entity can have attributes like Roll no, Name, and DeptID. They might have relationships with Courses and Lecturers.

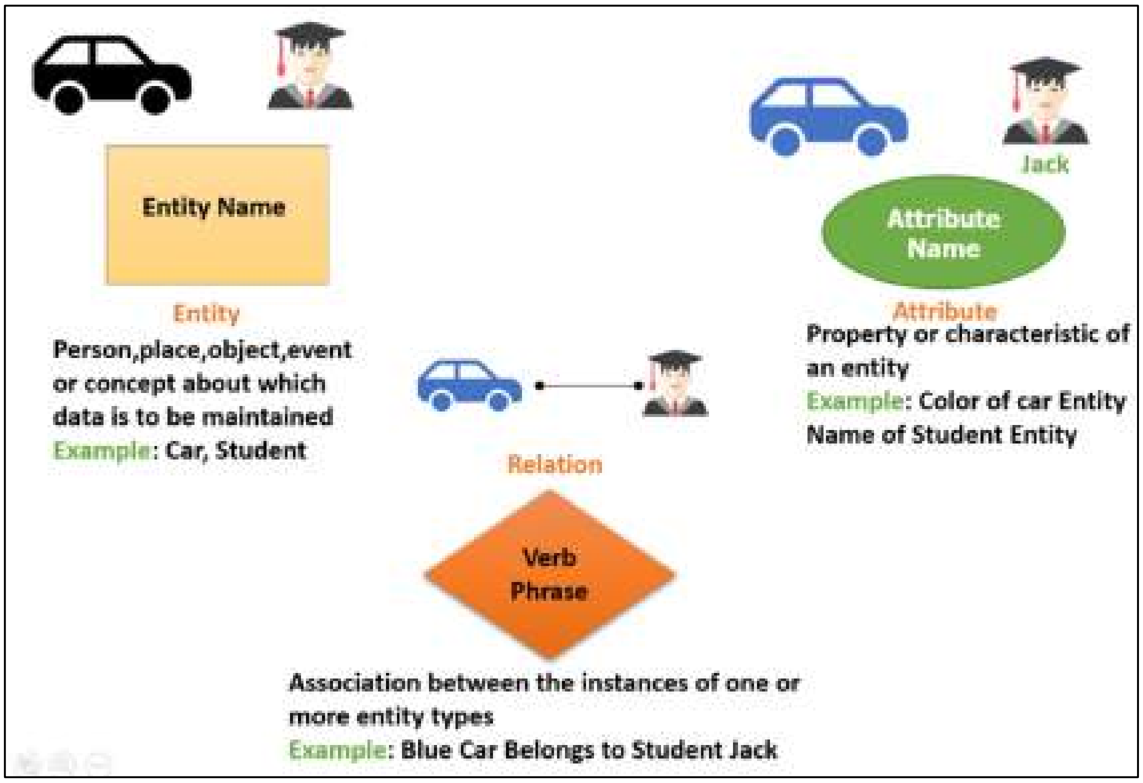

Entity Name
Entity
Person,place,object,event or concept about which data is to be maintained
Example: Car, Student
Jack
Attribute Name
Attribute
Property or characteristic of an entity
Example: Color of car Entity Name of Student Entity
Relation
Verb Phrase
Association between the instances of one or more entity types
Example: Blue Car Belongs to Student Jack

Figure 16: ER Diagram Example

3.6.2.2.1 ENTITIES

A real-world thing either living or non-living that is easily recognizable and nonrecognizable. It is anything in the enterprise that is to be represented in our database. It may be a physical thing or simply a fact about the enterprise or an event that happens in the real world.

An entity can be place, person, object, event or a concept, which stores data in the database. The characteristics of entities are must have an attribute, and a unique key. Every entity is made up of some ‘attributes’ which represent that entity.

Examples of entities:

- **Person:** Employee, Student, Patient
- **Place:** Store, Building
- **Object:** Machine, product, and Car
- **Event:** Sale, Registration, Renewal
- **Concept:** Account, Course

3.6.2.2.2 ATTRIBUTES

It is a single-valued property of either an entity-type or a relationship-type .For example, a lecture might have attributes: time, date, duration, place, etc.

An attribute in ER Diagram examples, is represented by an Ellipse

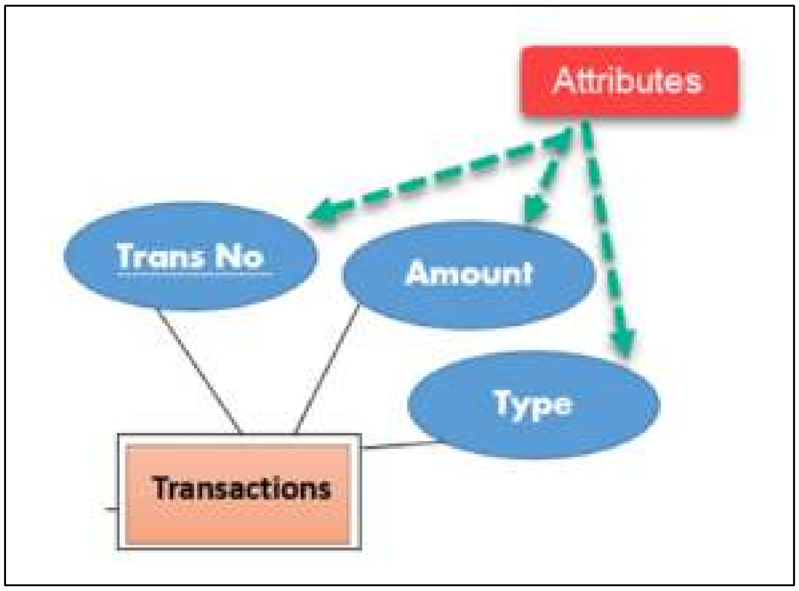


Figure 17: Attributes

3.6.2.2.3 RELATIONSHIPS OR CARDINALITY

Defines the numerical attributes of the relationship between two entities or entity sets.

Different types of cardinal relationships are:

- One-to-One Relationships
- One-to-Many Relationships
- May to One Relationships
- Many-to-Many Relationships

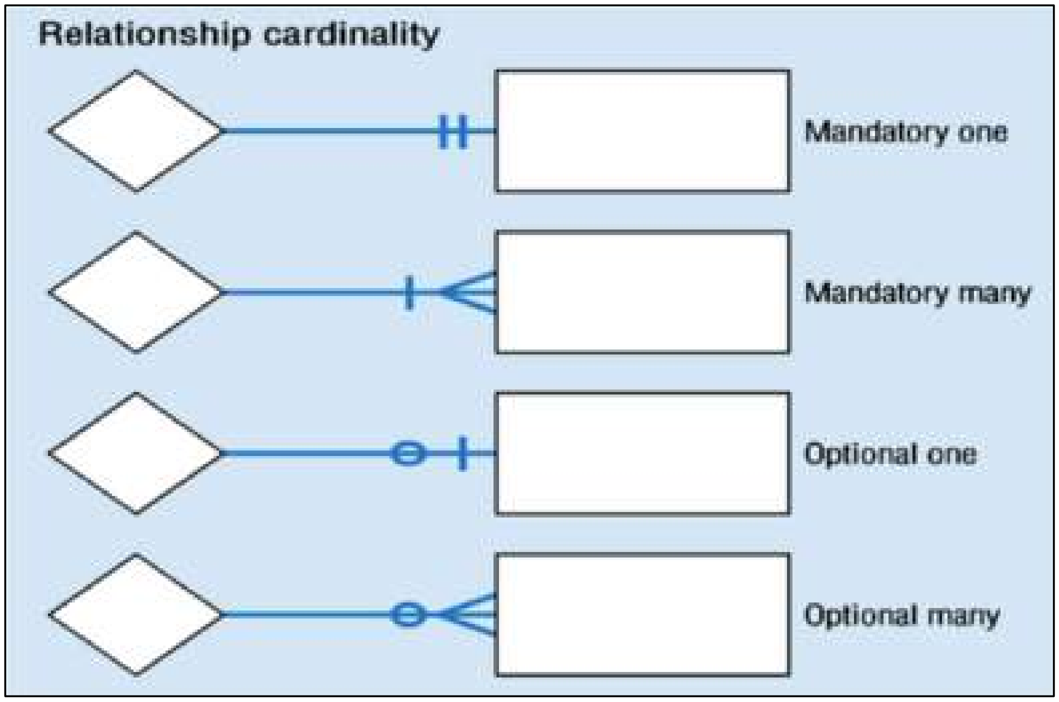


Figure 18: Relationship cardinality

3.6.2.3 ER DIAGRAM FOR SMART PARKING AND RESERVATIONS SYSTEM

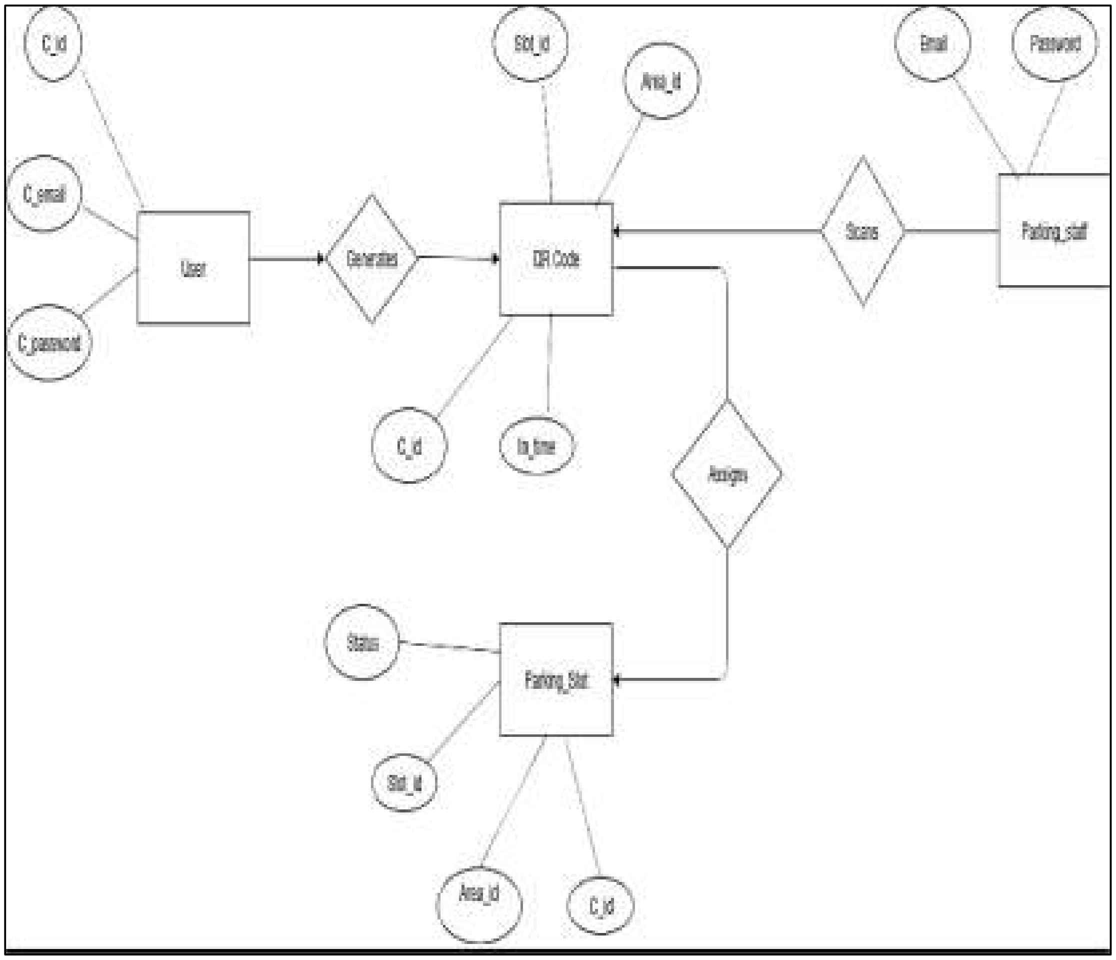


Figure 19:ER Diagram

3.6.3 USE CASE DIAGRAM

In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

- Scenarios in which your system or application interacts with people, organizations, or external systems
- Goals that your system or application helps those entities (known as actors) achieve
- The scope of your system

A use case diagram doesn't go into a lot of detail—for example, don't expect it to model the order in which steps are performed. Instead, a proper use case diagram depicts a high-level overview of the relationship between use cases, actors, and systems. Experts recommend that use case diagrams be used to supplement a more descriptive textual use case.

UML is the modelling toolkit that you can use to build your diagrams. Use cases are represented with a labelled oval shape. Stick figures represent actors in the process, and the actor's participation in the system is modelled with a line between the actor and use case. To depict the system boundary, draw a box around the use case itself.

UML use case diagrams are ideal for:

- Representing the goals of system-user interactions
- Defining and organizing functional requirements in a system
- Specifying the context and requirements of a system
- Modelling the basic flow of events in a use case

3.6.3.1 USE CASE DIAGRAM SYMBOLS AND NOTATION

The notation for a use case diagram is pretty straightforward and doesn't involve as many types of symbols as other UML diagrams. You can use this guide to learn how to draw a use case diagram if you need a refresher. Here are all the shapes you will be able to find in Lucid chart:

- **Use cases:** Horizontally shaped ovals that represent the different uses that a user might have.
- **Actors:** Stick figures that represent the people actually employing the use cases.
- **Associations:** A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.
- **System boundary boxes:** A box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system. For example, Psycho Killer is outside the scope of occupations in the chainsaw example found below.
- **Packages:** A UML shape that allows you to put different elements into groups. Just as with component diagrams, these groupings are represented as file folders.

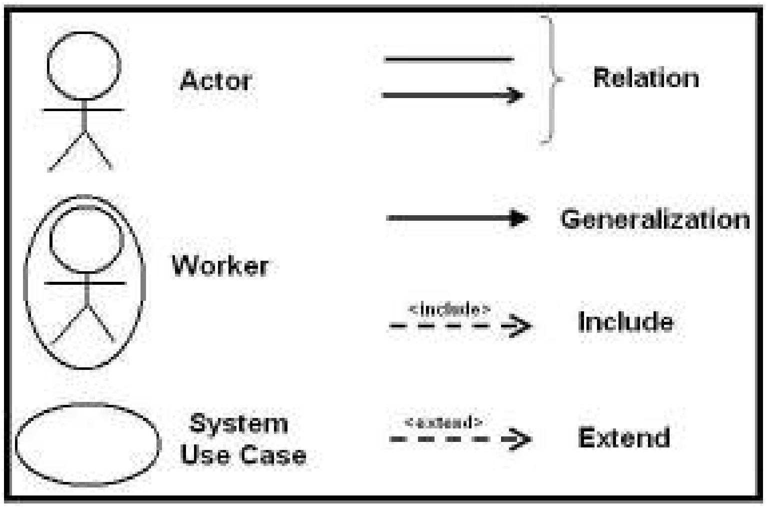


Figure 20: Use Case Diagram and Notation

3.6.3.2 USE CASE DIAGRAM FOR SMART PARKING AND RESERVATION SYSTEM

- **ADMIN USE CASE DIGRAM**

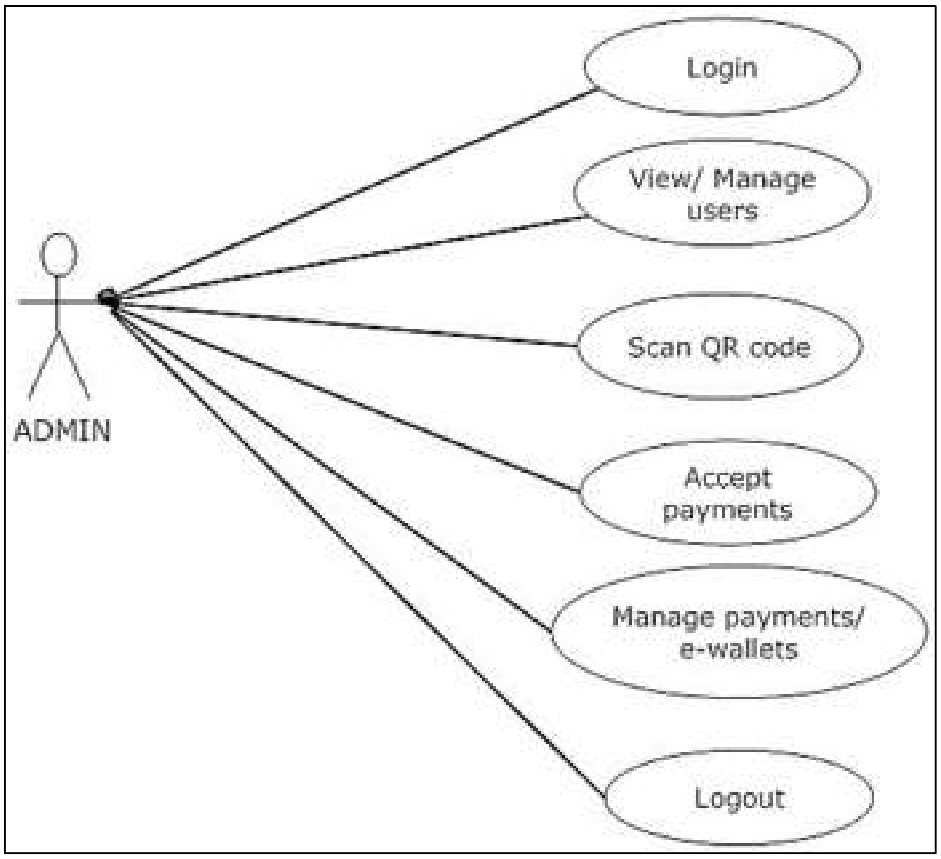


Figure 21: Admin Use Case Diagram

- **USER USE CASE DIAGRAM**

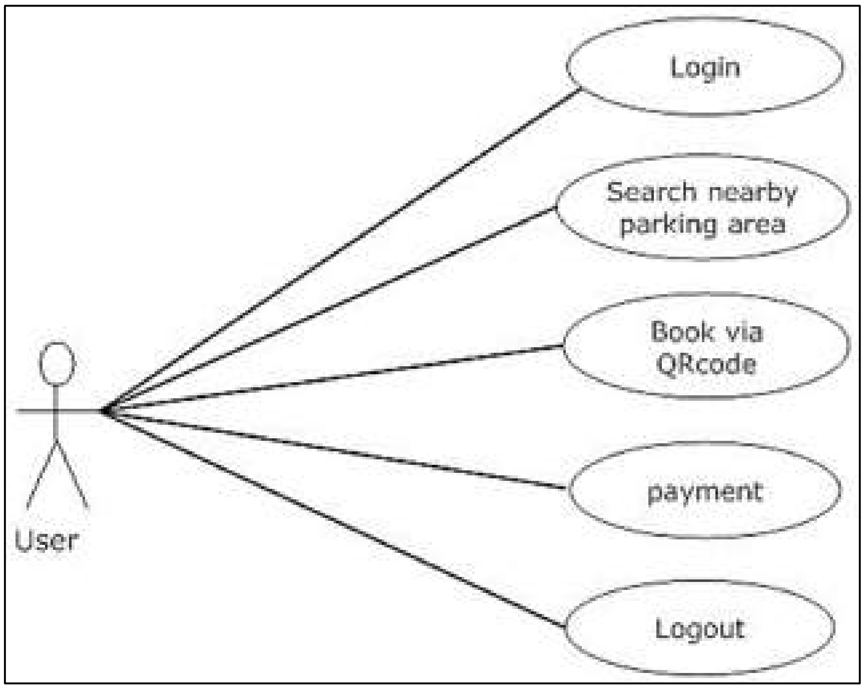


Figure 22: User use Case Diagram

3.6.4 SEQUENCE DIAGRAM

The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time. In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page. It incorporates the iterations as well as branching.

Purpose of a Sequence Diagram

1. To model high-level interaction among active objects within a system.
2. To model interaction among objects inside a collaboration realizing a use case.
3. It either model's generic interactions or some certain instances of interaction.

3.6.4.1 NOTATION OF A SEQUENCE DIAGRAM

1. Lifeline

An individual participant in the sequence diagram is represented by a lifeline. It is positioned at the top of the diagram.

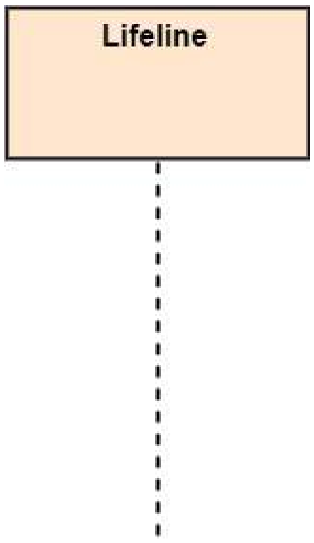


Figure 23: Sequence Diagram of Lifeline

2. Actor

A role played by an entity that interacts with the subject is called as an actor. It is out of the scope of the system. It represents the role, which involves human users and external hardware or subjects. An actor may or may not represent a physical entity, but it purely depicts the role of an entity. Several distinct roles can be played by an actor or vice versa.

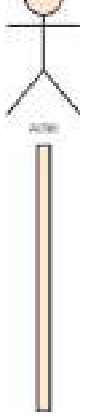


Figure 24: Sequence Diagram of Actor

3. Activation

It is represented by a thin rectangle on the lifeline. It describes that time period in which an operation is performed by an element, such that the top and the bottom of the rectangle is associated with the initiation and the completion time, each respectively.

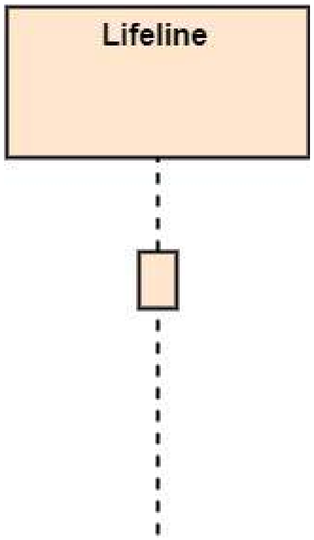


Figure 25: Sequence Diagram of Activation

4. Messages

The messages depict the interaction between the objects and are represented by arrows. They are in the sequential order on the lifeline. The core of the sequence diagram is formed by messages and lifelines.

Following are types of messages enlisted below:

- **Call Message:** It defines a particular communication between the lifelines of an interaction, which represents that the target lifeline has invoked an operation.

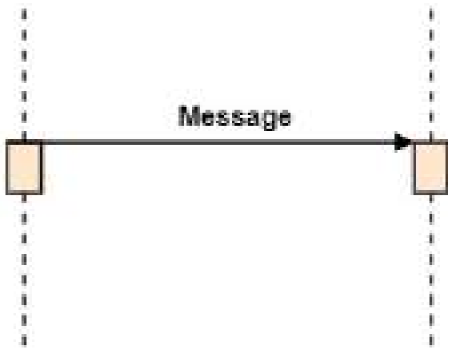


Figure 26: Sequence Diagram of Call Message

- **Return Message:** It defines a particular communication between the lifelines of interaction that represent the flow of information from the receiver of the corresponding caller message.

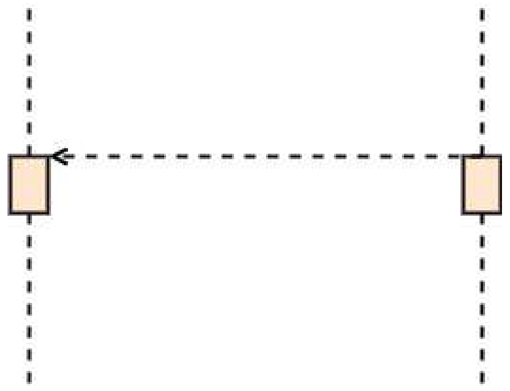


Figure 27: Sequence Diagram of Return Message

- **Self-Message:** It describes a communication, particularly between the lifelines of an interaction that represents a message of the same lifeline, has been invoked.

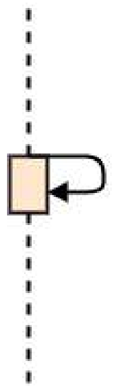


Figure 28: Sequence Diagram of Self Message

- **Recursive Message:** A self-message sent for recursive purpose is called a recursive message. In other words, it can be said that the recursive message is a special case of the self-message as it represents the recursive calls.

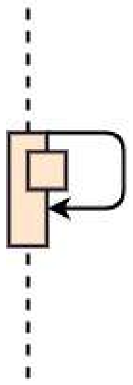


Figure 29: Sequence Diagram of Recursive Message

- **Create Message:** It describes a communication, particularly between the lifelines of an interaction describing that the target (lifeline) has been instantiated.

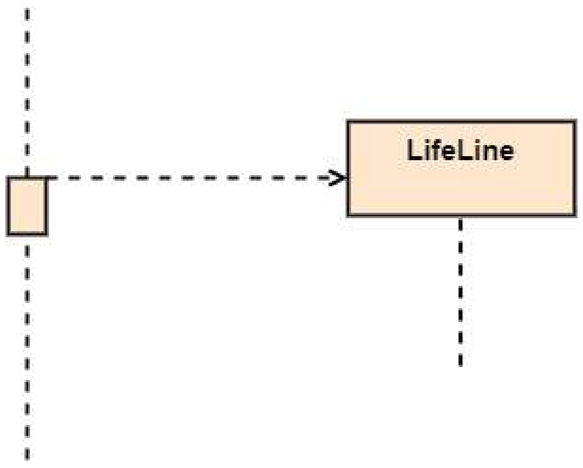


Figure 30: Sequence Diagram of Create Message

- **Destroy Message:** It describes a communication, particularly between the lifelines of an interaction that depicts a request to destroy the lifecycle of the target.

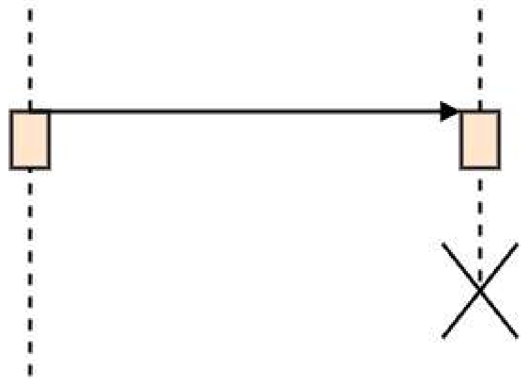


Figure 31: Sequence Diagram of Destroy Message

- **Duration Message:** It describes a communication particularly between the lifelines of an interaction, which portrays the time passage of the message while modelling a system.

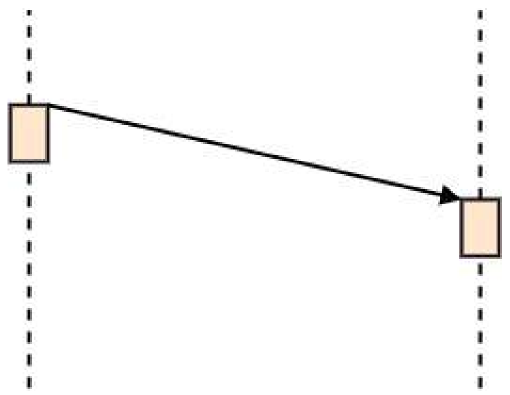


Figure 32: Sequence Diagram of Duration Message

3.6.4.2 BENIFITS OF SEQUENCE DIAGRAM

1. It explores the real-time application.
2. It depicts the message flow between the different objects.
3. It has easy maintenance.
4. It is easy to generate.
5. Implement both forward and reverse engineering.
6. It can easily update as per the new change in the system.

3.6.4.3 DRAWBACKS OF SEQUENCE DIAGRAM

1. In the case of too many lifelines, the sequence diagram can get more complex.
2. The incorrect result may be produced, if the order of the flow of messages changes.
3. Since each sequence needs distinct notations for its representation, it may make the diagram more complex.
4. The type of sequence is decided by the type of message.

3.6.4.4 SEQUENCE DIAGRAM OF SMART PARKING AND RESERVATION SYSTEM

- **ADMIN SEQUENCE DIAGRAM**

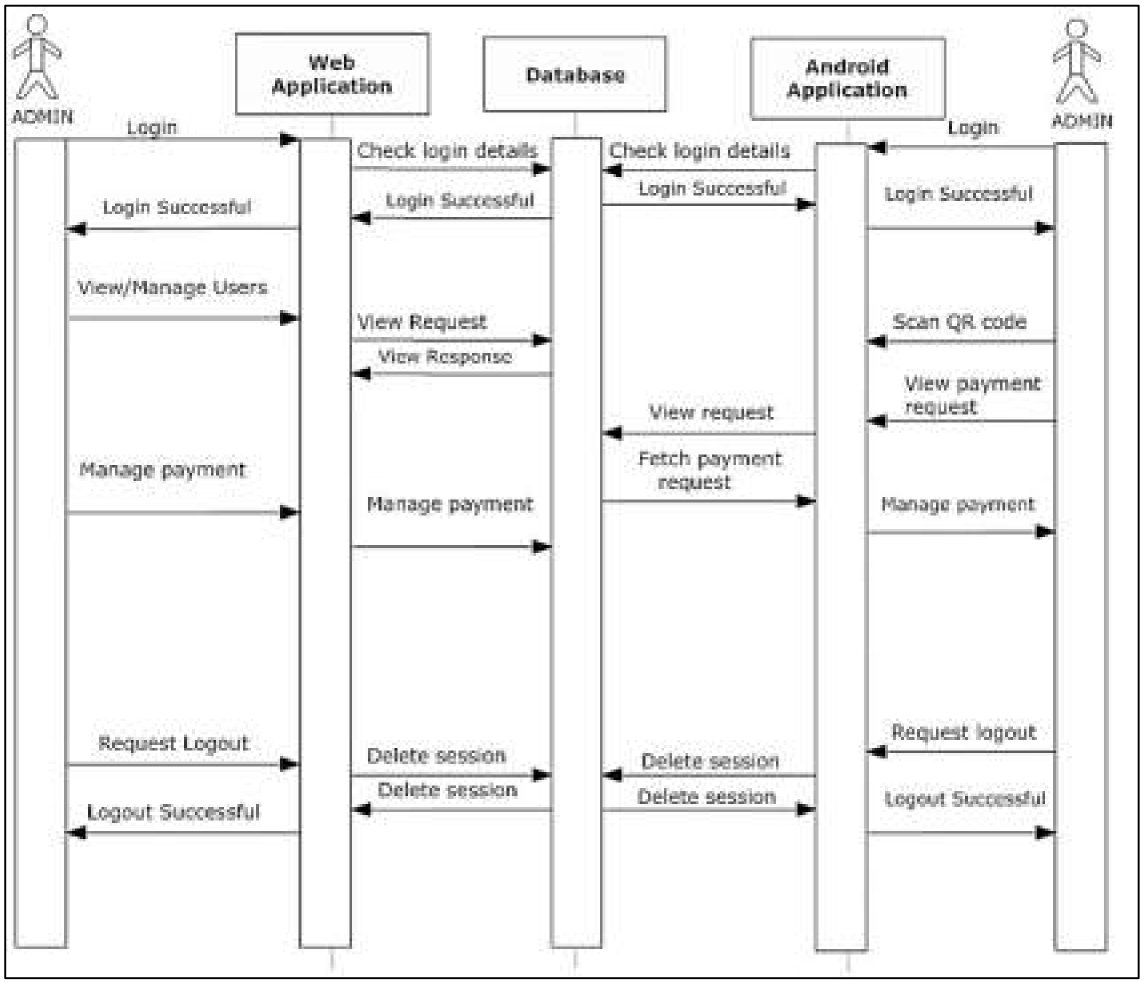


Figure 33: Admin Sequence Diagram

- **USER SEQUENCE DIAGRAM**

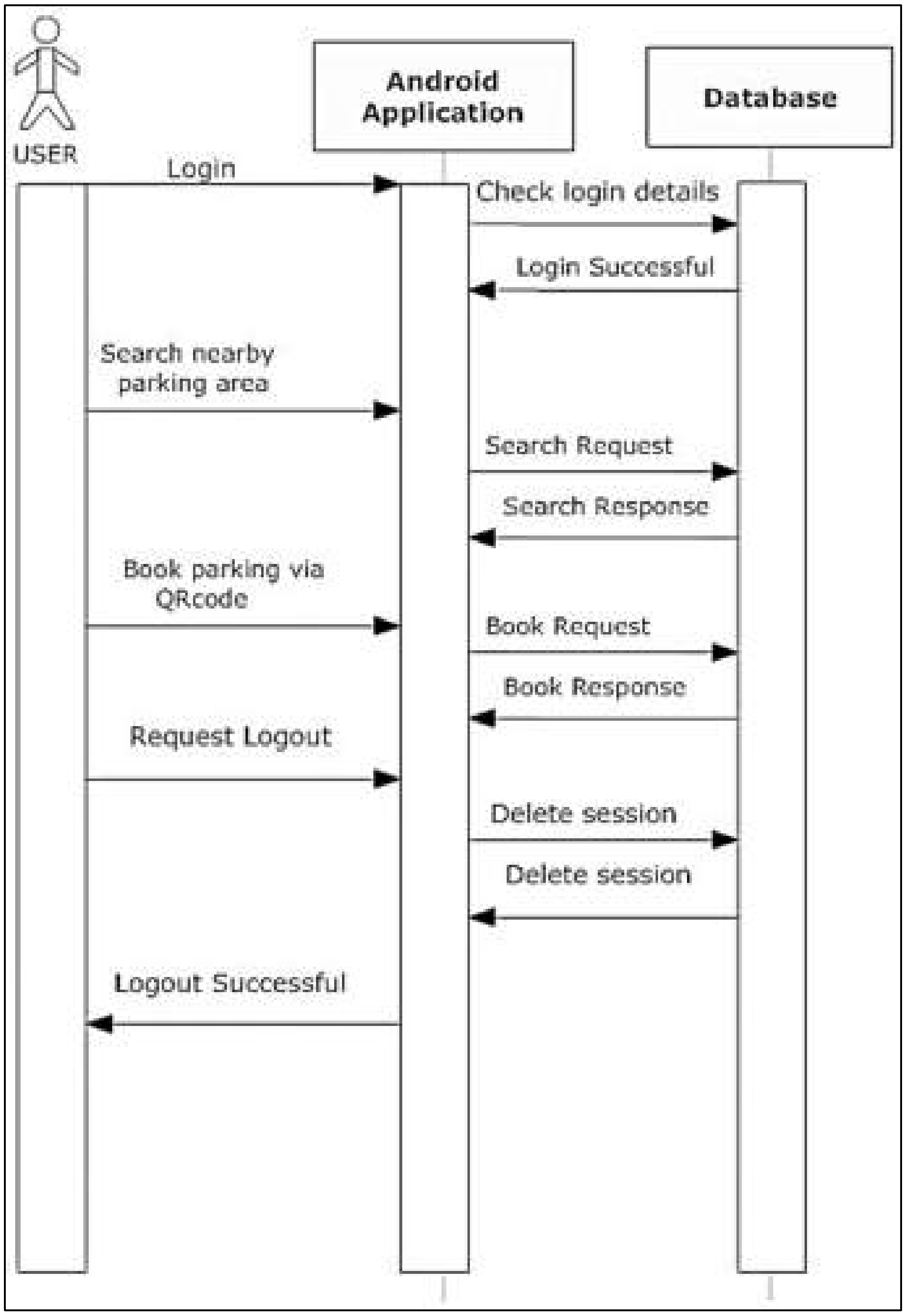


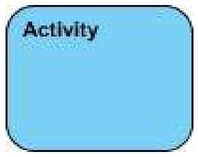
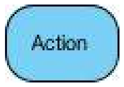
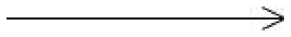
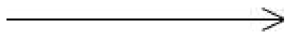
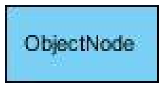
Figure 34: User Sequence Diagram

3.6.5 ACTIVITY DIAGRAM

Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single use case relate to one another, in particular, use cases where activities may overlap and require coordination. It is also suitable for modelling how a collection of use cases coordinates to represent business workflows

1. Identify candidate use cases, through the examination of business workflows.
2. Identify pre- and post-conditions (the context) for use cases.
3. Model workflows between/within use cases.
4. Model complex workflows in operations on objects.
5. Model in detail complex activities in a high-level activity Diagram.

3.6.5.1 ACITVITY DIAGRAM NOTATION

Notation Description	**UML Notation**
Activity Is used to represent a set of actions.	
Action A task to be performed.	
Control Flow Shows the sequence of execution	
Object Flow Show the flow of an object from one activity (or action) to another activity (or action).	
Initial Node Portrays the beginning of a set of actions or activities.	●
Activity Final Node Stop all control flows and object flows in an activity (or action).	◉
Object Node Represent an object that is connected to a set of Object Flows.	

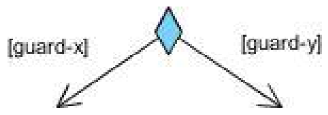
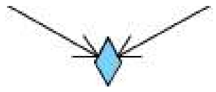
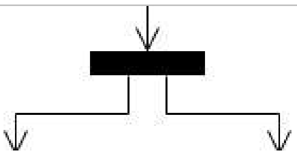
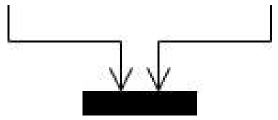
Decision Node Represent a test condition to ensure that the control flow or object flow only goes down one path.	
Merge Node Bring back together different decision paths that were created using a decision-node.	
Fork Node Split behaviour into a set of parallel or concurrent flows of activities (or actions).	
Join Node Bring back together a set of parallel or concurrent flows of activities (or actions).	

Figure 35: Activity Diagram Notation

3.6.5.2 ACITVITY DIAGRAM FOR SMART AND RESERVATION SYSTEM

- **ADMIN ACTIVITY DIAGRAM**

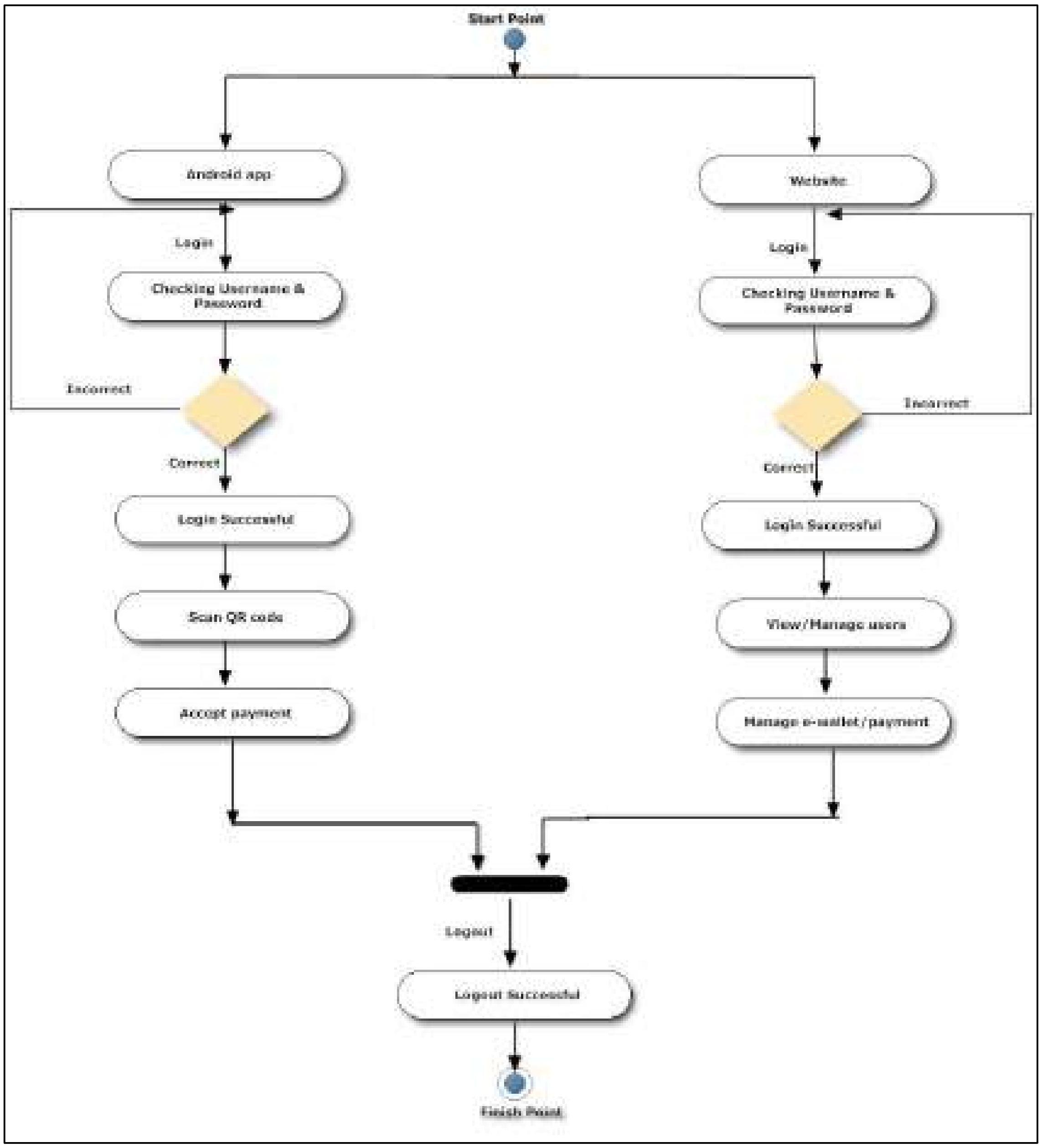


Figure 36: Admin Activity Diagram

- **USER ACTIVITY DIAGRAM**

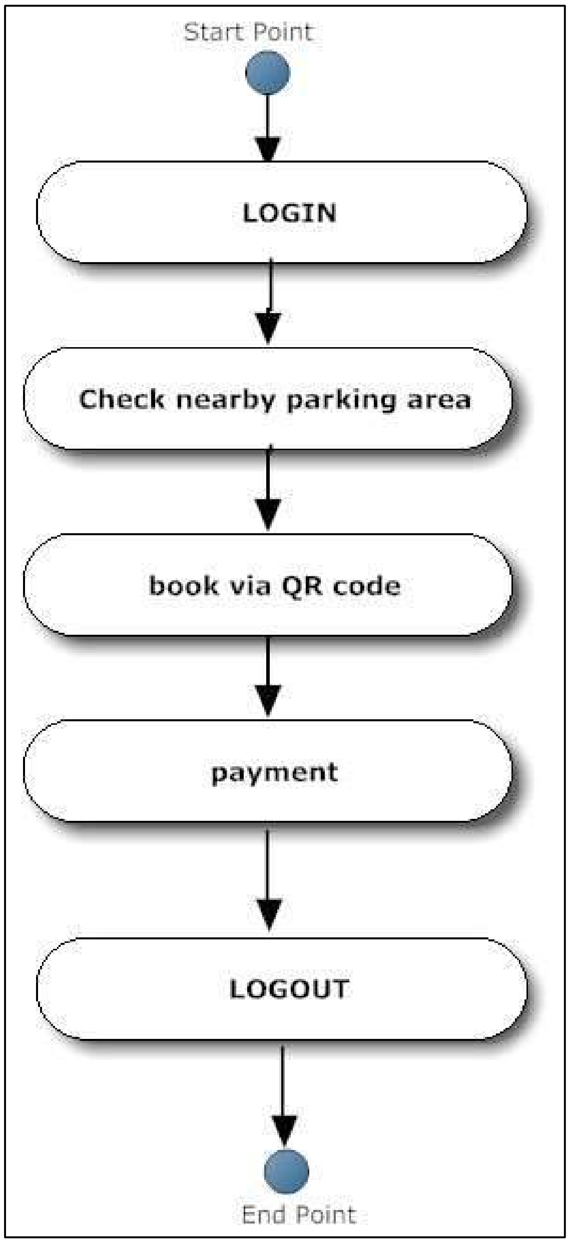


Figure 37: User Activity Diagram

3.6.6 UML COMPONENT DIAGRAM

component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node.

It visualizes the relationships as well as the organization between the components present in the system. It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behaviour is explained by the provided and required interfaces.

3.6.6.1 PURPOSE OF COMPONENT DIAGRAM

Since it is a special kind of a UML diagram, it holds distinct purposes. It describes all the individual components that are used to make the functionalities, but not the functionalities of the system. It visualizes the physical components inside the system. The components can be a library, packages, files, etc.

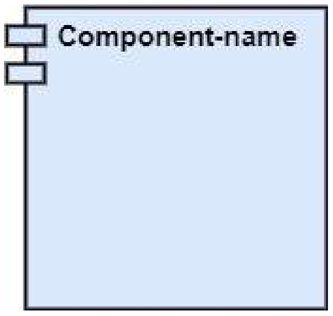
The component diagram also describes the static view of a system, which includes the organization of components at a particular instant. The collection of component diagrams represents a whole system.

The main purpose of the component diagram is enlisted below:

1. It envisions each component of a system.
2. It constructs the executable by incorporating forward and reverse engineering.
3. It depicts the relationships and organization of components.

3.6.6.2 NOTATION OF COMPONENT DIAGRAM

a) A component



b) A node

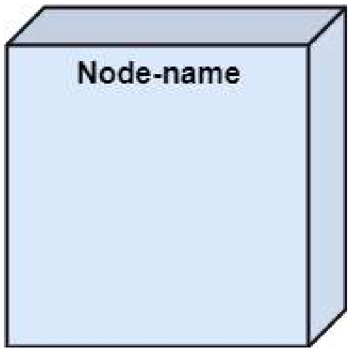


Figure 38: Notation of A node

3.6.6.3 COMPONENT DIAGRAM

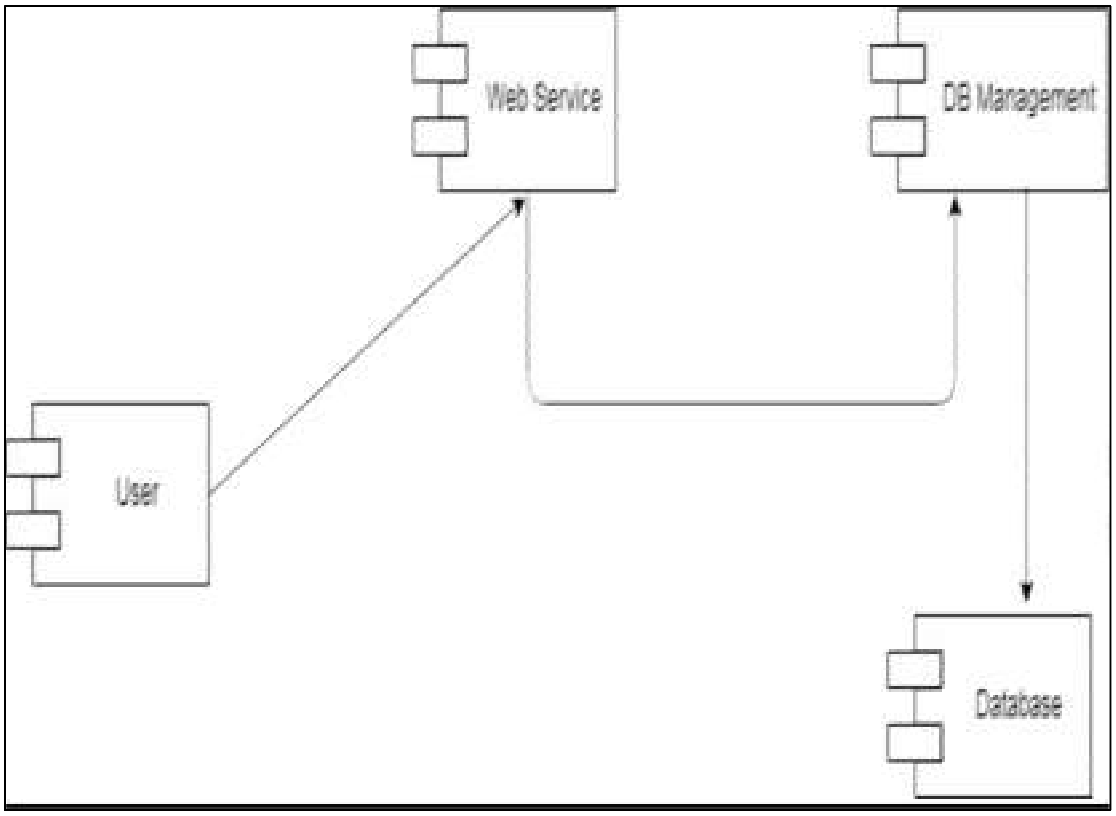


Figure 39: Component Diagram

CHAPTER 4

SYSTEM DESIGN

4.1 METHODOLOGIES USED FOR TESTING

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs (errors or other defects).

Software testing can also be stated as the process of validating and verifying that a software program/application/product:

- Meets the business and technical requirements that guided its design and development
- Works as expected; and
- Can be implemented with the same characteristics.

PRIMARY PURPOSE:

Testing is to detect software failures so that defects may be discovered and corrected. This is a non-trivial pursuit. Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions.

SCOPE:

The scope of software testing often includes examination of code as well as execution of that code in various environments and conditions as well as examining the aspects of code: does it do what it is supposed to do and do what it needs to do. In the current culture of software development, a testing organization may be separate from the development team.

IMPLEMENTATION:

Software testing, depending on the testing method employed, can be implemented at any time in the development process. However, most of the test effort occurs after the requirements have been defined and the coding process has been completed. As such, the methodology of the test is governed by the software development methodology adopted.

4.2 SOFTWARE TESTING MODEL

V MODEL

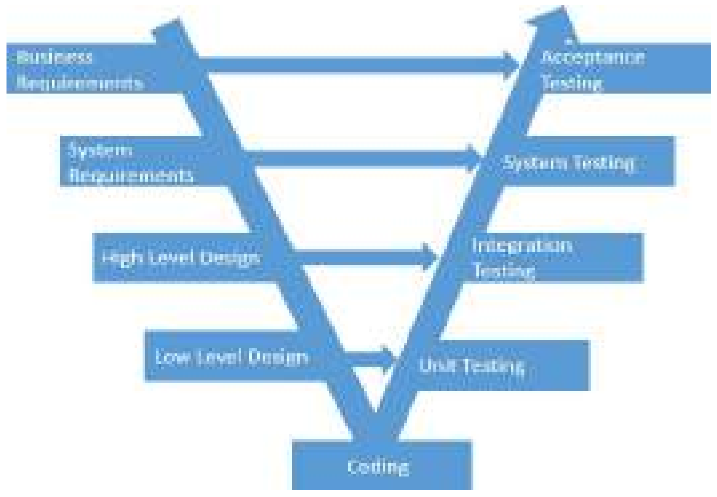


Figure 40: V-Model

The V-model involves building a logical V shape sequence where the testing techniques associated with the design are reflected as descending and are applied for the –verification‖ and connected to the requirements or specifications parts are reflected as ascending and are applied for –validation‖.

The V-model ordains that the code testing documentation is written in tandem with the development phases that means, for instance, the integration tests should be documented as and when the high-level design is finalized and the unit tests should be ready as and when the detailed specifications are laid down.

The idea of the V-model is to have an implementation plan for the software testing at each level namely component, interface, system, acceptance and release of the software project which need to be adhered to eliminate discrepancies in the software simultaneously rather than waiting for the software development process to complete before handling it to the software testing professionals.

4.3 DATA DICTIONARY

A Data Dictionary is a collection of names, definitions, and attributes about data elements that are being used or captured in a database, information system, or part of a research project. It describes the meanings and purposes of data elements within the context of a project, and provides guidance on interpretation, accepted meanings and representation. A Data Dictionary also provides metadata about data elements. The metadata included in a Data Dictionary can assist in defining the scope and characteristics of data elements, as well the rules for their usage and application.

Data Dictionaries are useful for a number of reasons. In short, they:

- Assist in avoiding data inconsistencies across a project
- Help define conventions that are to be used across a project
- Provide consistency in the collection and use of data across multiple members of a research team
- Make data easier to analyse
- Enforce the use of Data Standards

- Data Standards are rules that govern the way data are collected, recorded, and represented. Standards provide a commonly understood reference for the interpretation and use of data sets.

- By using standards, researchers in the same disciplines will know that the way their data are being collected and described will be the same across different projects. Using Data Standards as part of a well-crafted Data Dictionary can help increase the usability of your research data, and will ensure that data will be recognizable and usable beyond the immediate research team.

Area_master

Area_id	Int	Not null
Area_name	Varchar(50)	Not null
total_slot	Int	Not null
email_Id	Varchar(50)	Not null
Password	Varchar(50)	Not null
Lat	Varchar(50)	Not null
Lon	Varchar(50)	Not null

Table 2: Data Dictionary of Area Master

Booking_master

Booking_id	Int	Not null
Area_id	Int	Not null
Slot_id	Int	Not null
Status	Varchar(20)	Not null
Dt1	Datetime	Not null
Cust_id	Int	Not null
Dt2	Datetime	Not null
Cost	Varchar(20)	Not null

Table 3: Data Dictionary of Booking Master

User_master

C_id	Int	Not null
C_fname	Varchar(50)	Not null
C_lname	Varchar(50)	Not null
C_address	Varchar(50)	Not null
C_phone	Varchar(50)	Not null
C_email	Varchar(50)	Not null
C_password	Varchar(50)	Not null
C_balance	Int	Not null
Otp	Varchar(50)	Not null
Acc_status	Varchar(50)	Not null

Table 4: Data Dictionary of User Master

ParkingSlot_master

Slot_id	Int	Not Null
Area_id	Int	Not Null
Slot_no	Varchar(20)	Not Null
Flag	Varchar(20)	Not Null

Table 5: Data Dictionary of Parking Slot Master

4.4 DATA DESIGN

To understand how parking lots operate, we must know more about the types of people who visit parking lots. Customers who enter parking lots belong to one of the following groups:

- A regular customer who has purchased a biweekly, monthly, or yearly pass.
- A prepaid customer who booked a slot remotely (on the phone or online).
- A walk-in customer who neither has a pass nor booked a slot remotely. A slot will be assigned to such a customer based on availability.

Regular customers are usually given cards/stickers to place somewhere visible on their dashboard or windshield so the parking lot management can easily determine that the customers are not in violation of any parking rules. Unlike occasional visitors, regular customers are never issued parking slips on a daily basis. A parking lot typically reserves an entire block or floor for its regular visitors to ensure they always have places to park. Regular customers may also reserve slots for themselves so they can park their vehicles in the same designated slots every day, but this typically costs extra.

Those who make remote parking reservations may typically only use their designated slots for a limited time window of a couple hours, after which the slots are freed. When these visitors enter the parking lot, they must park in their reserved slots. A penalty is charged to customers who do not leave the parking lot after their time windows elapse, but customers can certainly leave before their reservations expire. Some parking lots have a fixed minimum time window (e.g., the customer may have to book a slot for three hours even if they are only going to be gone for one hour).

Walk-in customers are given parking slips when they enter a parking lot. A parking slot is then assigned to the customer as the slip is generated, based on preferences they have specified. The reservation process here is essentially the same as the one for prepaid customers. However, a walk-in reservation depends entirely on availability. A slot may cost you more than if you were to reserve a spot ahead of time, especially if there is limited availability and high demand.

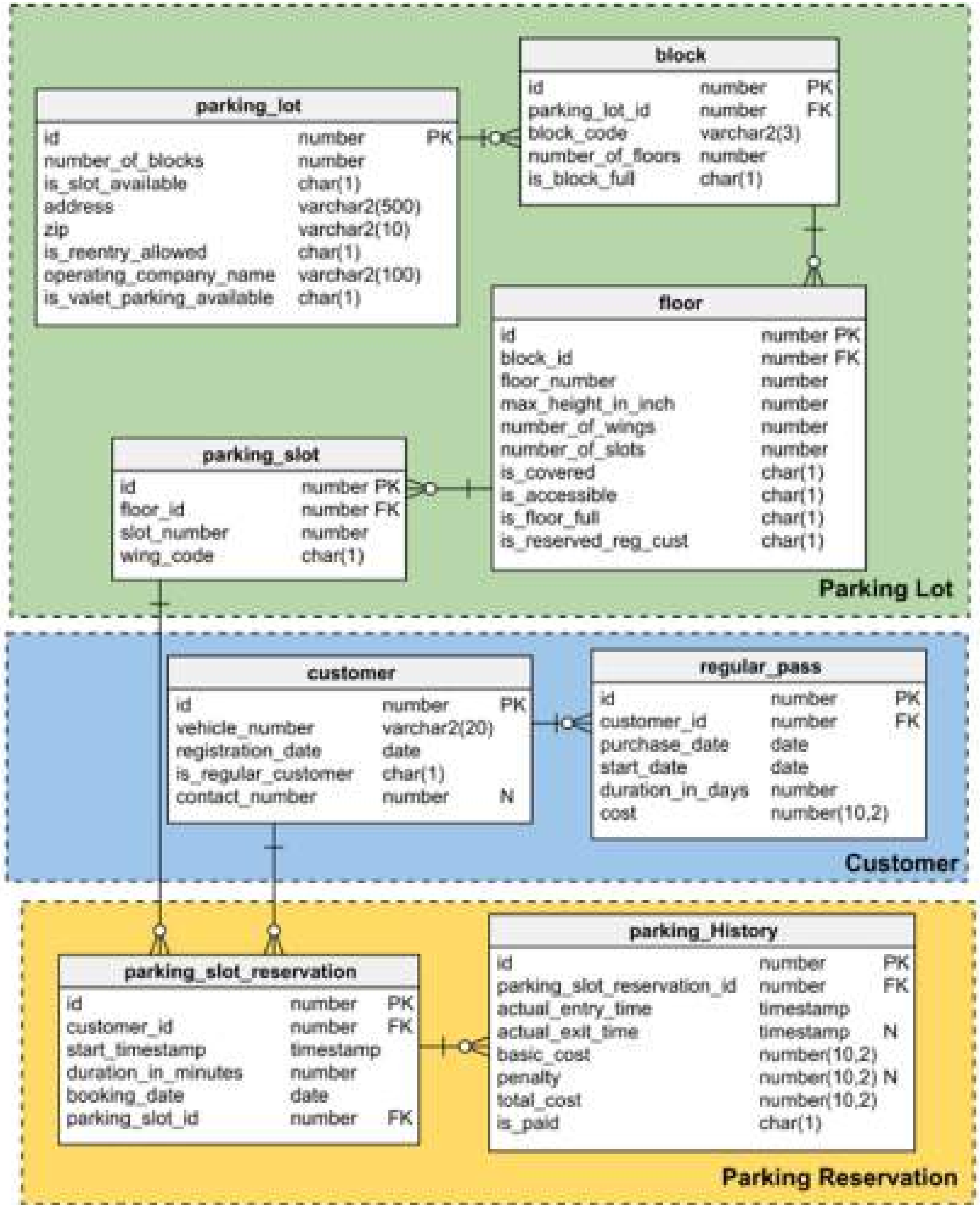


Figure 41: Data Design

4.4.1 SCHEMA DESIGN

4.4.1.1 CLASS DIAGRAM

A class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes,

their attributes, operations (or methods), and the relationships among objects. The class diagram is the main building block of object-oriented modelling.

It is used for general conceptual modelling of the systematic of the application, and for detailed modelling translating the models into programming code.

Class diagrams can also be used for data modelling. The classes in a class diagram represent both the main elements, interactions in the application, and the

classes to be programmed. In the diagram, classes are represented with boxes that contain three compartments:

I. The top compartment contains the name of the class. It is printed in bold and cantered, & the first letter is capitalized.
II. The middle compartment contains the attributes of the class. They are left-aligned the first letter is lowercase.
III. The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase.

A class with three compartments. In the design of a system, a number of classes are identified and grouped together in a class diagram that helps to determine the static relations between them. With detailed modelling, the classes of the conceptual design are often split into a number of subclasses.

CLASS DIAGRAM IMPLEMENTATION

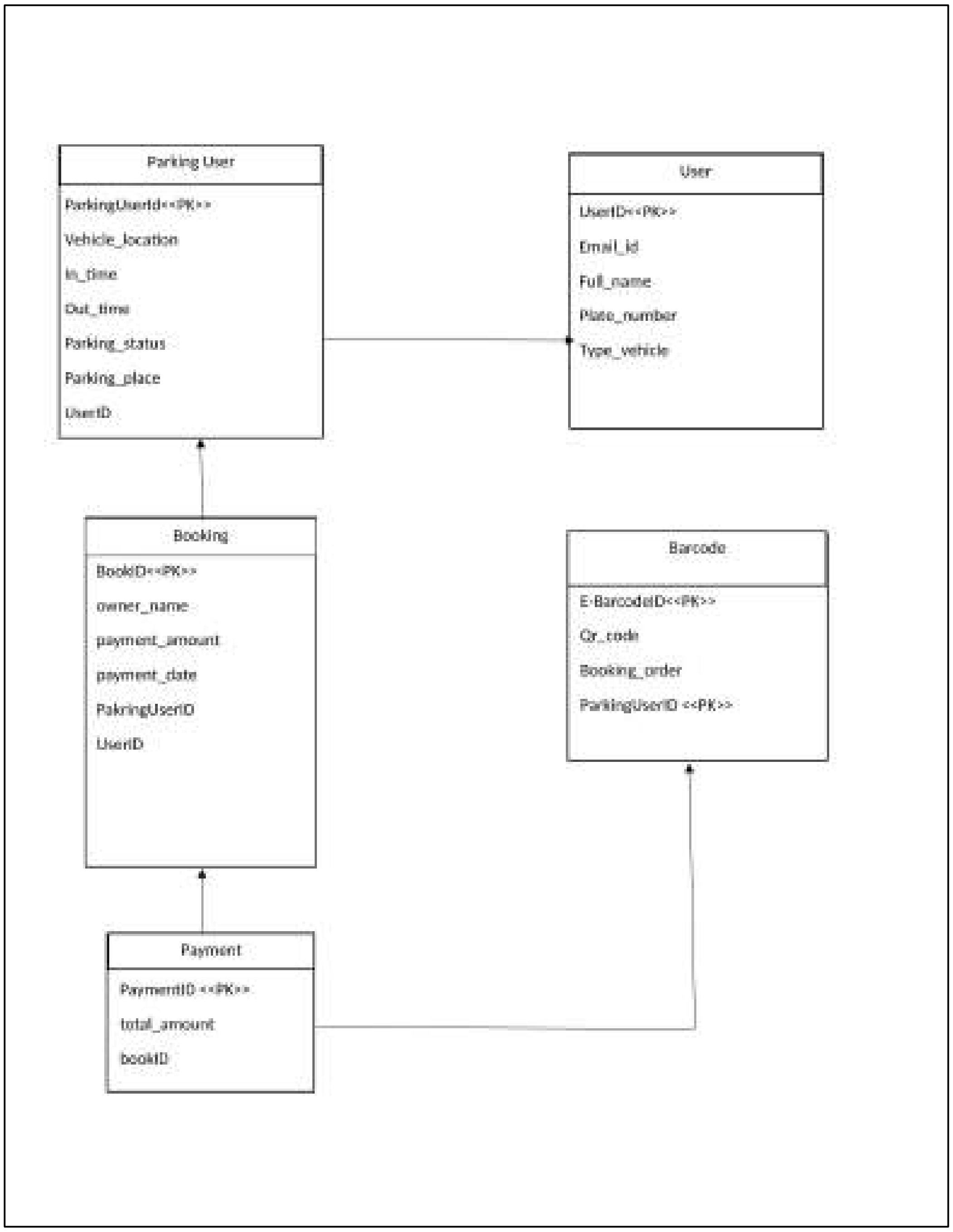


Figure 42: Class Diagram Implementation

4.4.2 DATA INTEGRITY AND CONSTRAINTS

Data Integrity and Constraints:

- .Primary Key Constraint - is a column or group of columns in a table that uniquely identify every row in that table

  .Foreign Key Constraint - is a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity.
- Check Constraint - Ensures that all values in a column satisfy certain conditions.
- Not NULL Constraint - It is Required because the User's username and password can't be null Because that's the only way, they have to authenticate himself/herself.
- Uniqueness Constraint - It is required because the Username columns shouldn't contain repeated value Every User will have unique username value to identify or authenticate himself/herself.

4.5 PROCEDURAL DESIGN

4.5.1 FLOW CHART

A project management process flowchart is a graphical aid, designed to visualize the sequence of steps to be followed throughout the project management process. Once your process flow has been developed, it will guide the primary phases of any future projects, from start to finish.

The purpose of any flowchart is to help visualize required steps – especially important for a project management process. Every flowchart consists of actions, the roles responsible for executing those actions and the inputs and outputs for each step. In addition, the flowchart will also include a record of any documents and other materials required to execute actions.

The goal of the process flowchart is clarity and transparency. The terminology used should be kept simple and free from unnecessary jargon – the steps won't be clear to new team members if they are full of confusing acronyms. For similar reasons, a flowchart convention should be agreed at the beginning (e.g., a square shape always stands for an action) and used consistently.

Once the project management process flowchart has mapped out the steps of each phase of the project and assigned ownership of responsibilities, everyone fully understands their role, and how they contribute to the whole.

FLOW CHART IMPLEMENTATION

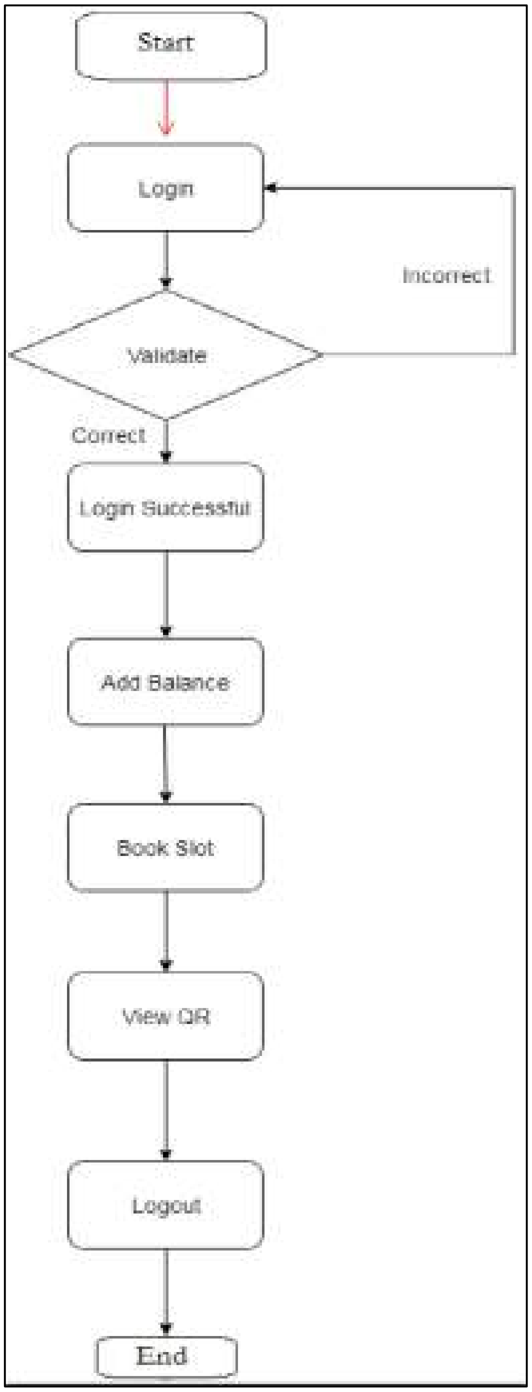


Figure 43: Flow Chart Implementation

4.6 USER INTERFACE DESIGN

1. **Start the application:**

The user needs to install the application on his Android based device. After installation, the icon of the app will feature on the Home Screen of the user's device. App welcome screen will be flashed to the user on opening the application.

2. **Registration:**

Initially, the user has to register his details with the application for the first time. This is a one-time registration. The user has to enter details like user name, gender, phone number and email- id. All this data will be stored on server. Booking for slots mandatory has to be done 45 min prior to arrival. On server side the parking owner also needs to register the number of parking slots available and for what type of vehicles and the amount that needsto be paid.

3. **Login:**

Once the user registers, he can use his email id and phone number to login in future. This authenticates the user.

4. **Selection of location for parking:**

The user is provided with multiple parking locations. User has to select one of the locations provided where he desires to park the vehicle.

5. **Select vehicle type:**

After selecting the location, options for the vehicle type are provided i.e., 2-wheeler or 4-wheeler alongside the rate chart for parking charges is prompted.

6. **Enter user's details for slot reservation and Money**

Figure 44: Menu of Smart Phone showing the smart phone application short cut icon marked

When clicked on the icon of the application the first page appears as shown in the figure 42 below which prompts the user to enter mobile number and password and then log in to the system if he is already registered else the user will have to click on registration button.

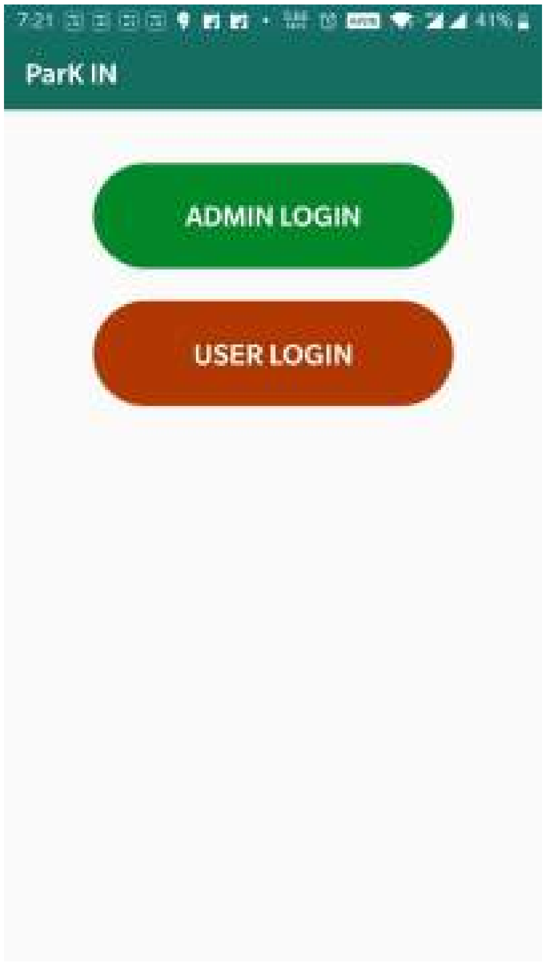


Figure 45: Login page of the application

After clicking on registration button, the user will have to register himself by entering the appropriate details into the fields asshown in the Figure 43.

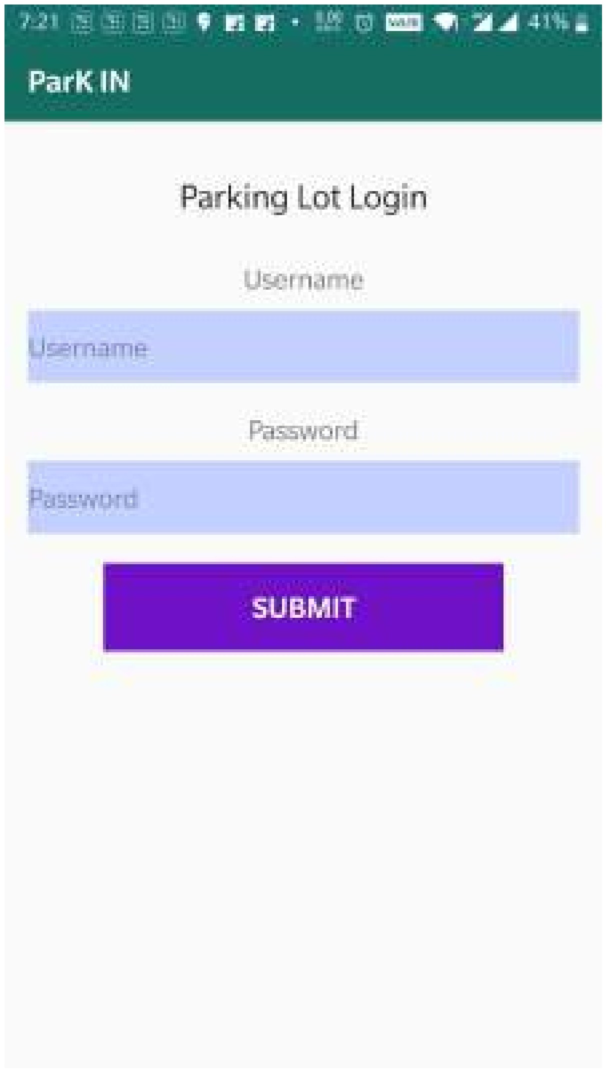


Figure 46: Registration page of the application

After submitting the details, the login page will appear with fields mobile number and password. User will have to enter correct mobile number and password same as he entered while registration. After successful login next page which contains the menu including options that are map, list, QR Code scanner and logout as shown in the figure will be displayed.

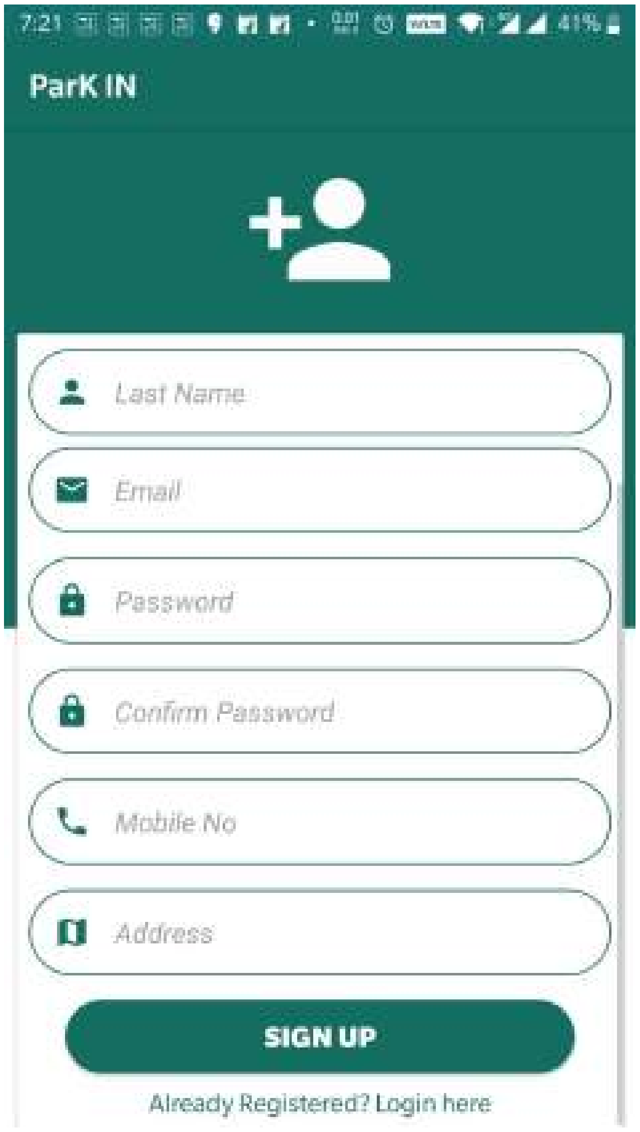


Figure 47: Main menu of the application

The user is provided with multiple parking locations. User has to select one of the locations provided where he desires to park the vehicle. After selecting the location, the remaining details including parking time(in time and out time),vehicle number, card expiry date, card holder name ,card number, and then vcc/cvv number and then the user has to do the payment

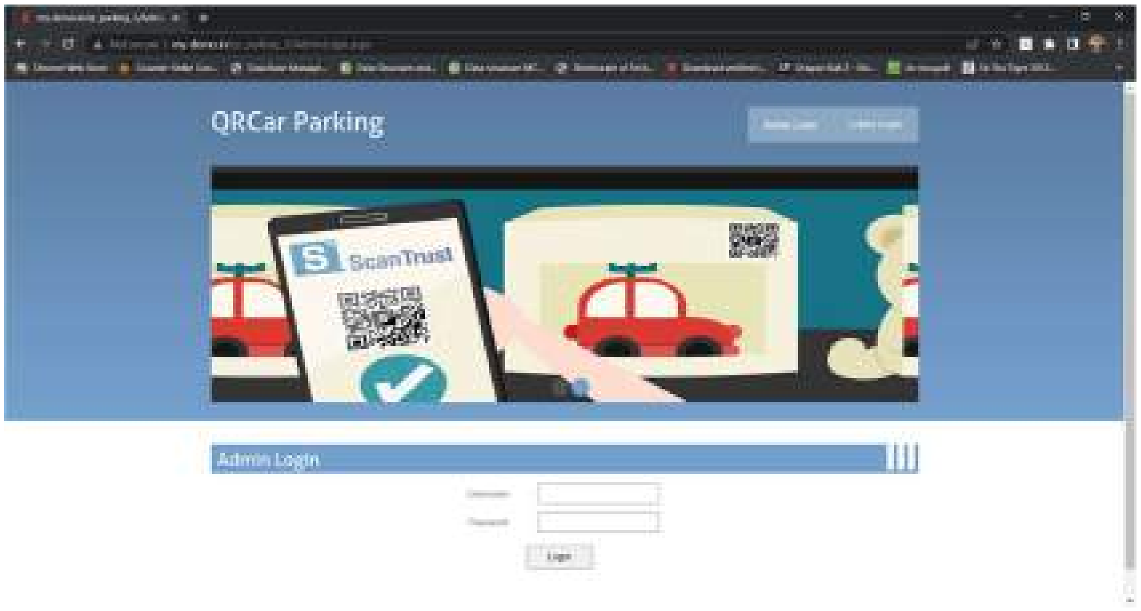


Figure 48: Admin Login Page

CHAPTER 5

IMPLEMENTATION AND TESTING

5.1 <u>IMPLEMENTATION APPROACH</u>

System testing is an essential step for the development of a reliable and error-free system. Once source code has been generated, software must be tested to uncover and correct as many errors as possible before delivery to your customer. Your goal is to design a series of test cases that have a high likelihood finding errors but how, there are different methods that provides a systematic guidance for designing tests that, Exercise the internal logic of software components, and exercise the input and output domains of the program to uncover errors in the program function, behavior, and performance.

Software testing is a crucial element of software quality assurance and represents the ultimate review of specification, design, and code generation. The work product is a set of test cases designed to exercise both internal logic and external requirements is designed and documented, expected results are defined, and actual results are recorded. The primary objectives of testing software are to execute a program with the intent of finding an error; a good test case will find an as-yet-undercover error, and a successful that uncover an as-yet-undercover error.

5.2 CODING DETAILS AND CODE EFFICIENCY

Website Code

- **Admin login code:-**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

public partial class _Default : System.Web.UI.Page
{
    SqlConnection conn;
    SqlCommand comm;
    SqlDataAdapter da;
    DataSet ds;
    public string cs=ConfigurationManager.AppSettings["Connect"].ToString();

    protected void Page_Load(object sender, EventArgs e)
    {
        if(Session["a_id"]!=null)
        {
            Session.Abandon();
            Session.Clear();
        }
    }
    protected void btnsubmit_Click(object sender, EventArgs e)
    {
        conn=new SqlConnection(cs);
        conn.Open();
        da = new SqlDataAdapter();
        da.SelectCommand=new SqlCommand();
        da.SelectCommand.Connection = conn;
        da.SelectCommand.CommandText="Ad_login";
        da.SelectCommand.CommandType=CommandType.StoredProcedure;
        da.SelectCommand.Parameters.AddWithValue("@username",txtusername.Text);
        da.SelectCommand.Parameters.AddWithValue("@password",txtpassword.Text);
        DataTable dt=new DataTable();
        da.Fill(dt);
        conn.Close();
        if(dt.Rows.Count>0)
        {
           Session["a_id"]=dt.Rows[0]["Ad_id"].ToString();
           Response.Redirect("Home.aspx");
        }
        else
        {
           Label1.Visible=true;
           Label1.Text="Invalid Details";
        }

    }
}
```

- **Lobby Login Code :-**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

public partial class _Default : System.Web.UI.Page
{
    SqlConnection conn;
    SqlCommand comm;
    SqlDataAdapter da;
    DataSet ds;
    public string cs = ConfigurationManager.AppSettings["Connect"].ToString();
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["l_id"] != null)
        {
            Session.Abandon();
            Session.Clear();
        }
    }
    protected void btnsubmit_Click(object sender, EventArgs e)
    {
        conn = new SqlConnection(cs);
        conn.Open();
        da = new SqlDataAdapter();
        da.SelectCommand = new SqlCommand();
        da.SelectCommand.Connection = conn;
        da.SelectCommand.CommandText = "Select * from Area_master where
email_ID=@username and Password=@password";
        da.SelectCommand.Parameters.AddWithValue("@username", txtusername.Text);
        da.SelectCommand.Parameters.AddWithValue("@password", txtpassword.Text);
        DataTable dt = new DataTable();
        da.Fill(dt);
        if (dt.Rows.Count > 0)
        {
            Session["l_id"] = dt.Rows[0]["Area_id"].ToString();
            Response.Redirect("AddBalance.aspx");
        }
else
{
   Label1.Visible = true;
   Label1.Text = "Invalid Details";
}}}
```

- **Manage Parking Code :-**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
using System.Net.Mail;
using System.IO;

public partial class _Default : System.Web.UI.Page
{
    SqlConnection conn;
    SqlDataAdapter adp;
    public string cs = ConfigurationManager.AppSettings["Connect"].ToString();
    DataTable dt;

    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            if (Session["a_id"] != null)
            {
                conn = new SqlConnection(cs);
                conn.Open();
                adp = new SqlDataAdapter("select Area_id,Area_name,total_slot from Area_master",
conn);
                dt = new DataTable();
                adp.Fill(dt);
                if (dt.Rows.Count > 0)
                {
                    GridView1.DataSource = dt;
                    GridView1.DataBind();
                }
                else
                {
                    Label1.Visible = true;
                    Label1.Text = "No record Found";
                }
                conn.Close();
            }
            else
            {
                Response.Write("<script>alert('Log in first')</script>");
                Response.Redirect("AdminLogin.aspx");
            }

        }
    }

    protected void btnadd_Click(object sender, EventArgs e)
    {

        try
        {
            if (FileUpload1.HasFile)
            {
                string fileName = Path.GetFileName(FileUpload1.FileName);
                FileUpload1.SaveAs(Server.MapPath("Parking_Images/") + fileName);
                string filePath = "/Parking_Images/" + fileName;

                conn = new SqlConnection(cs);
                //Generate Random Password
                var chars =
"QWERTYUIOPLKJHGFDSAZXCVBNMqwertyuioplkjhgfdsazxcvbnm0987654321";
                var stringargs = new char[8];
                var random = new Random();
                for (int i = 0; i < stringargs.Length; i++)
                {
                    stringargs[i] = chars[random.Next(chars.Length)];
                }
                string password = new String(stringargs);

                SqlCommand _cmd = new SqlCommand("Insert_area", conn);
                _cmd.CommandType = CommandType.StoredProcedure;
                _cmd.Parameters.AddWithValue("@area_name", txtarea.Text);
                _cmd.Parameters.AddWithValue("@total_slots", txtttlslots.Text);
                _cmd.Parameters.AddWithValue("@mail", txtMail.Text);
                _cmd.Parameters.AddWithValue("@pass", password);
                _cmd.Parameters.AddWithValue("@lat", txtLat.Text);
                _cmd.Parameters.AddWithValue("@lon", txtLon.Text);
                _cmd.Parameters.AddWithValue("@img", filePath);
                conn.Open();
                _cmd.ExecuteNonQuery();
                conn.Close();
                sms1(txtMail.Text, password);
            Response.Redirect("Manage_parking.aspx"); Response.Write("<script>alert('Added
Successfully')</script>");
          }
          else
            Response.Write("<script>alert('Please upload image')</script>");
        }
        catch(Exception ex)
        {
          throw ex;
        }
    }

    public void sms1(string mailid, string password)
    {

        MailMessage mail = new MailMessage();
        SmtpClient SmtpServer = new SmtpClient("my-demo.in");
        mail.From = new MailAddress("test@my-demo.in");
        mail.To.Add(mailid);
        mail.Subject = "Registration Details";
        mail.Body = "Your Login-id is : " + mailid + "\n" + "Your Password is : " + password;
        SmtpServer.Port = 25;
        SmtpServer.Credentials = new System.Net.NetworkCredential("test@my-demo.in",
"Password@123");
        SmtpServer.Send(mail);

    }

    public void Row_command(object sender, GridViewCommandEventArgs e)
    {

        if (e.CommandName == "Delete_row")
        {
          int index = Convert.ToInt32(e.CommandArgument);
          GridViewRow gr = GridView1.Rows[index];
          conn = new SqlConnection(cs);
          conn.Open();
          SqlCommand cmd2 = new SqlCommand("delete from Area_master where Area_id='" +
gr.Cells[0].Text + "'", conn);
          cmd2.ExecuteNonQuery();
          conn.Close();
          Response.Write("<script>alert('Deleted')</script>");
          Response.Redirect("Manage_parking.aspx");
        }
}}
```

- **Manage Customer Code :-**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

public partial class _Default : System.Web.UI.Page
{
    SqlConnection conn;
    SqlDataAdapter adp;
    public string cs = ConfigurationManager.AppSettings["Connect"].ToString();
    DataTable dt;
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            if (Session["a_id"] != null)
            {
                conn = new SqlConnection(cs);
                conn.Open();
                adp = new SqlDataAdapter("select C_id,C_fname,C_ph,C_email,C_balance from
Customer_master", conn);
                dt = new DataTable();
                adp.Fill(dt);
                if (dt.Rows.Count > 0)
                {
                    GridView1.DataSource = dt;
                    GridView1.DataBind();
                }
                else
                {
                    Label1.Visible = true;
                    Label1.Text = "No record Found";
                }
                conn.Close();
            }
            else
            {
                Response.Write("<script>alert('Log in first')</script>");
                Response.Redirect("AdminLogin.aspx");
            }
        }
    }

    public void Row_command(object sender, GridViewCommandEventArgs e)
    {

        if (e.CommandName == "Delete_row")
        {
            int index = Convert.ToInt32(e.CommandArgument);
            GridViewRow gr = GridView1.Rows[index];
            conn = new SqlConnection(cs);
            conn.Open();
            SqlCommand cmd2 = new SqlCommand("delete from Customer_master where C_id='" +
gr.Cells[0].Text + "'", conn);
            cmd2.ExecuteNonQuery();
            conn.Close();
            Response.Write("<script>alert('Deleted')</script>");
            Response.Redirect("ManageCustomers.aspx");
        }
    }
}
```

Android Code

- **Login activity code :-**

```
package com.example.cp.qr_car_parking_app;

import android.app.Dialog;
import android.content.Intent;
import android.os.Handler;
import android.os.Message;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.example.cp.qr_car_parking_app.Connection.ConnectionM;
import com.example.cp.qr_car_parking_app.Connection.Progressdialog;

public class LoginActivity extends AppCompatActivity {

    Dialog dg;
    int resp;
    TextView tvGoToSignup,txtForgot;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        tvGoToSignup = findViewById(R.id.tvGotoSignup);
        txtForgot = findViewById(R.id.txtForgot);

        Button btnlogin = (Button) findViewById(R.id.btnlogin);
        btnlogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Login();
            }
        });
        tvGoToSignup.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent=new Intent(LoginActivity.this,Register.class);
                startActivity(intent);
            }
        });

        txtForgot.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                forgeet();
            }
        });

    }

    public void forgeet()
    {
        try {
            EditText txtuser = (EditText) findViewById(R.id.editmail);
            final String usermail = txtuser.getText().toString();

            if (usermail.equals("") ) {
                Toast.makeText(this, "Enter Email Id", Toast.LENGTH_LONG).show();
            } else {
                final ConnectionM conn = new ConnectionM();
                if (ConnectionM.checkNetworkAvailable(this)) {
                    Progressdialog dialog = new Progressdialog();
                    dg = dialog.createDialog(this);
                    dg.show();

                    Thread th1 = new Thread() {
                        @Override
                        public void run() {
                            try {
                                if (conn.forget(usermail)) {
                                    resp = 0;
                                } else {
                                    resp = 1;
                                }
                            } catch (Exception e) {
                                e.printStackTrace();
                            }
                            hd2.sendEmptyMessage(0);
                        }
                    };
                    th1.start();
                } else {
                    Toast.makeText(this, "Sorry no network access.",
    Toast.LENGTH_LONG).show();
                }
            }
        } catch (Exception e) {

        }
    }

    public Handler hd2 = new Handler() {
        public void handleMessage(Message msg) {
            dg.cancel();
            switch (resp) {
                case 0:
                    Toast.makeText(getApplicationContext(), "Password is sent to your email id",
    Toast.LENGTH_LONG).show();
                    break;

                case 1:
                    Toast.makeText(getApplicationContext(), "Invalid Email or Try Later",
    Toast.LENGTH_LONG).show();

                    break;
            }
        }
    };

    public void Login() {
        try {
            EditText txtuser = (EditText) findViewById(R.id.editmail);
            EditText txtpass = (EditText) findViewById(R.id.editpass);
            final String usermail = txtuser.getText().toString();
            final String userpass = txtpass.getText().toString();
            if (usermail.equals("") || userpass.equals("")) {
                Toast.makeText(this, "all fields are mandatory", Toast.LENGTH_LONG).show();
            } else {
                final ConnectionM conn = new ConnectionM();
                if (ConnectionM.checkNetworkAvailable(this)) {
                    Progressdialog dialog = new Progressdialog();
                    dg = dialog.createDialog(this);
                    dg.show();

                    Thread th1 = new Thread() {
                        @Override
                        public void run() {
                            try {
                                if (conn.authenticate_user(usermail, userpass)) {
                                    resp = 0;
                                } else {
                                    resp = 1;
                                }
                            } catch (Exception e) {
                                e.printStackTrace();
                            }
                            hd.sendEmptyMessage(0);
                        }
                    };
                    th1.start();
                } else {
                    Toast.makeText(this, "Sorry no network access.",
        Toast.LENGTH_LONG).show();
                }
            }
        } catch (Exception e) {

        }
      }

      public Handler hd = new Handler() {
          public void handleMessage(Message msg) {
              dg.cancel();
              switch (resp) {
                  case 0:
                      EditText txtemailid = (EditText) findViewById(R.id.editmail);
                      EditText txtpassword = (EditText) findViewById(R.id.editpass);

                      txtemailid.setText("");
                      txtpassword.setText("");

                      Intent intent = new Intent(LoginActivity.this, Main_.class);
                      startActivity(intent);
                      break;

                  case 1:

                      EditText txtemailid1 = (EditText) findViewById(R.id.editmail);
                      EditText txtpassword1 = (EditText) findViewById(R.id.editpass);

                      txtemailid1.setText("");
```

```
                    txtpassword1.setText("");

                    Toast.makeText(getApplicationContext(), "Invalid Email Id Or Password",
        Toast.LENGTH_LONG).show();

                    break;
                }
            }
        };

    }
```

- **Parking Lot Login code :-**

```
package com.example.cp.qr_car_parking_app;

import android.app.Dialog;
import android.content.Intent;
import android.os.Handler;
import android.os.Message;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.example.cp.qr_car_parking_app.Connection.ConnectionM;
import com.example.cp.qr_car_parking_app.Connection.Progressdialog;

public class ParkingLotLogin extends AppCompatActivity {

    Dialog dg;
    int resp;

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

        Button btnPL=(Button)findViewById(R.id.btnPLLogin);
        btnPL.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                login();
            }
        });

    }

    public void login()
    {
        try {
            EditText txtuser = (EditText) findViewById(R.id.editPLmail);
            EditText txtpass = (EditText) findViewById(R.id.editPLpass);
            final String usermail = txtuser.getText().toString();
            final String userpass = txtpass.getText().toString();
            if (usermail.equals("") || userpass.equals("")) {
                Toast.makeText(this, "all fields are mandatory", Toast.LENGTH_LONG).show();
            } else {
                final ConnectionM conn = new ConnectionM();
                if (ConnectionM.checkNetworkAvailable(this)) {
                    Progressdialog dialog = new Progressdialog();
                    dg = dialog.createDialog(this);
                    dg.show();

                    Thread th1 = new Thread() {
                        @Override
                        public void run() {
                            try {
                                if (conn.authPL(usermail, userpass)) {
                                    resp = 0;
                                } else {
                                    resp = 1;
                                }
                            } catch (Exception e) {
                                e.printStackTrace();
                            }
                            hd.sendEmptyMessage(0);
                        }
                    };
                    th1.start();
                } else {
                    Toast.makeText(this, "Sorry no network access.",
        Toast.LENGTH_LONG).show();
                }
            }
        } catch (Exception e) {

        }
    }

    public Handler hd = new Handler() {
        public void handleMessage(Message msg) {
            dg.cancel();
            switch (resp) {
                case 0:
                    EditText txtpassword = (EditText) findViewById(R.id.editPLpass);
                    txtpassword.setText("");

                    Intent intent = new Intent(ParkingLotLogin.this, ParkingLot_Home.class);
                    startActivity(intent);
                    break;
                case 1:

                    EditText txtpassword1 = (EditText) findViewById(R.id.editPLpass);
                    txtpassword1.setText("");

                    Toast.makeText(getApplicationContext(), "Invalid Email Id Or Password",
Toast.LENGTH_LONG).show();

                    break;
            }
        }
    };



}
```

- **Main page code :-**

```
package com.example.cp.qr_car_parking_app;

import android.Manifest;
import android.app.AlertDialog;
import android.app.Dialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
import android.location.Location;

import com.example.cp.qr_car_parking_app.Connection.ConnectionM;
import com.example.cp.qr_car_parking_app.Connection.Progressdialog;
import com.example.cp.qr_car_parking_app.Data.Area_data;
import com.example.cp.qr_car_parking_app.Data.Cust_data;
import com.example.cp.qr_car_parking_app.Data.Loc_point;
import com.example.cp.qr_car_parking_app.Data.TransLog;
import com.google.android.gms.common.api.GoogleApiClient;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.common.api.ResultCallback;
import com.google.android.gms.common.api.Status;
import com.google.android.gms.location.LocationListener;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;


import org.w3c.dom.Text;
import java.util.ArrayList;

public class Main_ extends AppCompatActivity implements
        GoogleApiClient.ConnectionCallbacks,
        GoogleApiClient.OnConnectionFailedListener,
        LocationListener,
        ResultCallback<Status>{

    Dialog dg;
    int resp;

    ProgressDialog progressDialog;

    private GoogleApiClient googleApiClient;
    private Location lastLocation;
    ImageView imgPark,imgQr,imgTrans,imgBal,imgFeed,imgLog;

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

        createGoogleApi();

        //TODO  need to be start after LoginActivity
        imgPark = findViewById(R.id.imgPark);
        imgPark.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(Main_.this, View_Parking.class);
                startActivity(intent);
            }
        });

        imgQr = findViewById(R.id.imgQr);
        imgQr.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Intent intent=new Intent(Main_.this,View_QR.class);
                // startActivity(intent);
                Check_Book_info();
            }
        });

        imgTrans = findViewById(R.id.imgTrans);
        imgTrans.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                getTransLog();
            }
        });


        imgBal = findViewById(R.id.imgBal);
        imgBal.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent=new Intent(Main_.this,AddBalance.class);
                startActivity(intent);
            }
        });

        TextView txtBal=(TextView)findViewById(R.id.txtCustBalance);
        txtBal.setText("Balance: "+Cust_data.getBal());

        imgFeed = findViewById(R.id.imgFeed);
        imgFeed.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent=new Intent(Main_.this,Feedback.class);
                startActivity(intent);
            }
        });

        imgLog = findViewById(R.id.imgLogout);
        imgLog.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                logout();
            }
        });

    }

    public void Check_Book_info() {
        final ConnectionM conn = new ConnectionM();
        if (ConnectionM.checkNetworkAvailable(this)) {
            Progressdialog dialog = new Progressdialog();
```

```
            dg = dialog.createDialog(this);
            dg.show();

            Thread th1 = new Thread() {
                @Override
                public void run() {
                    try {
                        if (conn.getBookedLog()) {
                            resp = 0;
                        } else {
                            resp = 1;
                        }
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                    hd.sendEmptyMessage(0);
                }
            };
            th1.start();
        } else {
            Toast.makeText(this, "Sorry no network access.", Toast.LENGTH_LONG).show();
        }
    }

    public Handler hd = new Handler() {
        public void handleMessage(Message msg) {
            dg.cancel();
            switch (resp) {
                case 0:
                    Intent intent = new Intent(Main_.this, View_QR.class);
                    startActivity(intent);
                    break;

                case 1:
                    Toast.makeText(getApplicationContext(), "Booking Details Not found",
Toast.LENGTH_LONG).show();
                    break;
            }
        }
    };

    public void getTransLog()
    {
        final ConnectionM conn = new ConnectionM();
        if (ConnectionM.checkNetworkAvailable(this)) {
            Progressdialog dialog = new Progressdialog();
            dg = dialog.createDialog(this);
            dg.show();

            Thread tthread = new Thread() {
                @Override
                public void run() {
                    try {
                        if (conn.getTransLog()) {
                            resp = 0;
                        } else {
                            resp = 1;
                        }
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                    hd2.sendEmptyMessage(0);

                }
            };
            tthread.start();
        } else {
            Toast.makeText(getApplicationContext(), "Sorry no network access.",
    Toast.LENGTH_LONG).show();
        }
    }

    public Handler hd2 = new Handler() {
        public void handleMessage(Message msg) {
            dg.cancel();
            switch (resp) {
                case 0:

                    ArrayList<String> id = new ArrayList<String>();
                    id = TransLog.getBid();
                   if(!id.isEmpty())
                   {
                      Intent intent=new Intent(Main_.this,TransactionLog.class);
                      startActivity(intent);
                   }
                    else
                   {
                      Toast.makeText(Main_.this, "Data not found",
    Toast.LENGTH_SHORT).show();
                   }
                    break;
                case 1:
                    Toast.makeText(getApplicationContext(), "data not received",
        Toast.LENGTH_LONG).show();

                    break;
            }
        }
    };


    @Override
    public void onBackPressed() {
        logout();
    }

    public void logout() {
        new android.support.v7.app.AlertDialog.Builder(Main_.this)
                .setIcon(R.drawable.alert)
                .setTitle(R.string.app_name)
                .setMessage("Are you sure you want logout?")
                .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        Intent intent = new Intent(Main_.this,LoginActivity.class);

                        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                        startActivity(intent);
                        finish();
                    }

                })
                .setNegativeButton("No", null)
                .show();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
        try
        {
            MenuInflater inflater=getMenuInflater();
            inflater.inflate(com.example.cp.qr_car_parking_app.R.menu.view_qr, menu);
            return true;
        }
        catch(Exception ex)
        {
            String msg=ex.getLocalizedMessage();
            return false;
        }
    }


    @Override
    public boolean onOptionsItemSelected(MenuItem item)
    {
        try {
            switch (item.getItemId())
            {
                case R.id.mnuQR:
                    Check_Book_info();
                    break;
                default:
                    return super.onOptionsItemSelected(item);
            }
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return false;
    }


    private void createGoogleApi() {
        //Log.d(TAG, "createGoogleApi()");
        if (googleApiClient == null) {
            googleApiClient = new GoogleApiClient.Builder(this)
                    .addConnectionCallbacks(this)
                    .addOnConnectionFailedListener(this)
                    .addApi(LocationServices.API)
                    .build();
        }
        googleApiClient.connect();
    }

    private final int REQ_PERMISSION = 999;

    private boolean checkPermission() {
        // Ask for permission if it wasn't granted yet
        return (ContextCompat.checkSelfPermission(this,
android.Manifest.permission.ACCESS_FINE_LOCATION)
                == PackageManager.PERMISSION_GRANTED);
    }
```

```
private void askPermission() {
    ActivityCompat.requestPermissions(
            this,
            new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
            REQ_PERMISSION
    );
}

@Override
public void onConnectionSuspended(int i) {
    //Log.w(TAG, "onConnectionSuspended()");
}

// GoogleApiClient.OnConnectionFailedListener fail
@Override
public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
    //Log.w(TAG, "onConnectionFailed()");
}

@Override
public void onLocationChanged(Location location) {
    //Log.d(TAG, "onLocationChanged ["+location+"]");
    lastLocation = location;

    Loc_point.setLat(location.getLatitude());
    Loc_point.setLon(location.getLongitude());
    //writeActualLocation(location);
}

@Override
public void onConnected(@Nullable Bundle bundle) {
    //Log.i(TAG, "onConnected()");
    getLastKnownLocation();
}

@Override
public void onResult(@NonNull Status status) {
    // Log.i(TAG, "onResult: " + status);
    if (status.isSuccess()) {

    } else {
        // inform about fail
    }
}
```

```
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {
        //Log.d(TAG, "onRequestPermissionsResult()");
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        switch (requestCode) {
            case REQ_PERMISSION: {
                if (grantResults.length > 0
                        && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                    // Permission granted
                    getLastKnownLocation();

                } else {
                    // Permission denied
                    Toast.makeText(Main_.this, "Failed !! \n Start GPS Service .....",
Toast.LENGTH_SHORT).show();
                }
                break;
            }
        }
    }

    private void getLastKnownLocation() {
        //Log.d(TAG, "getLastKnownLocation()");
        if (checkPermission()) {
            lastLocation = LocationServices.FusedLocationApi.getLastLocation(googleApiClient);
            if (lastLocation != null) {
                //Log.i(TAG, "LasKnown location. " +                        "Long: " +
lastLocation.getLongitude() +                        " | Lat: " + lastLocation.getLatitude());
                //writeLastLocation();
                startLocationUpdates();
            } else {
                //Log.w(TAG, "No location retrieved yet");
                startLocationUpdates();
            }
        } else askPermission();
    }

    private LocationRequest locationRequest;
    // Defined in mili seconds.
    // This number in extremely low, and should be used only for debug
    private final int UPDATE_INTERVAL = 1000;
    private final int FASTEST_INTERVAL = 900;


    private void startLocationUpdates() {
        //Log.i(TAG, "startLocationUpdates()");
        locationRequest = LocationRequest.create()
                .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY)
                .setInterval(UPDATE_INTERVAL)
                .setFastestInterval(FASTEST_INTERVAL);

        if (checkPermission())
            LocationServices.FusedLocationApi.requestLocationUpdates(googleApiClient,
locationRequest, this);
    }


}
```

- **View Parking Code :-**

```
package com.example.cp.qr_car_parking_app;

import android.app.Dialog;
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.Spinner;
import android.widget.Toast;

import com.example.cp.qr_car_parking_app.Connection.ConnectionM;
import com.example.cp.qr_car_parking_app.Connection.Progressdialog;
import com.example.cp.qr_car_parking_app.Data.Area_data;
import com.example.cp.qr_car_parking_app.Data.SelectedArea;
import com.example.cp.qr_car_parking_app.Data.Slot_info;
import com.example.cp.qr_car_parking_app.Data.Slots_data;
import com.squareup.picasso.Picasso;

import java.util.ArrayList;

public class View_Parking extends AppCompatActivity {

    ProgressDialog dg1,dg2,dg3,dg4;
    int resp;
    static ArrayList<String> slot_no;
    static ArrayList<String> slot_id;
    static ArrayList<String> slot_url;

    public static ArrayList<String> lat;
    public static ArrayList<String> lon;
    public static ArrayList<String> name;
    public static ArrayList<String> area_img;
    Spinner spSortBy;
    ImageView imgArea;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_view__parking);
        Fill_Area();
        // ArrayList<String> Names = new ArrayList<String>();
        // Names = Area_data.getList_name();

        imgArea = findViewById(R.id.imgArea1);

        // if (Names.isEmpty()) {
        //     Toast.makeText(this, "Empty", Toast.LENGTH_LONG).show();
        // } else {
        spSortBy = (Spinner) findViewById(R.id.spSort);
        spSortBy.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
                String sortByName = spSortBy.getSelectedItem().toString();
                Area_data.setByName(sortByName);
                if (sortByName.equals("Rating")) {
                    Fill_AreaBy("Rating");
                } else if (sortByName.equals("Number of Slots")) {
                    Fill_AreaBy("nos");
                } else {
                    Fill_Area();
                }
            }

            @Override
            public void onNothingSelected(AdapterView<?> adapterView) {

            }
        });

        final Spinner sp = (Spinner) findViewById(R.id.spinnerarea);
//
        //     sp.setAdapter(new ArrayAdapter<String>(View_Parking.this,
android.R.layout.simple_spinner_dropdown_item, Names));

        sp.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {

                int test = i;
                String test2 = sp.getSelectedItem().toString();
                int a = 0;
                fillParkList(test2);

                SelectedArea.setAreaName(test2);
                SelectedArea.setLat(lat.get(i));
                SelectedArea.setLon(lon.get(i));



                //Toast.makeText(getApplicationContext(),"http://my-
    demo.in/qr_parking_3"+area_img.get(i),Toast.LENGTH_LONG).show();
            }

            @Override
            public void onNothingSelected(AdapterView<?> adapterView) {
                //TODO
            }
        });
        // }
        ListView lst = (ListView) findViewById(R.id.listView);
        lst.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
                String selectedSlot = slot_id.get(i);
                Slot_info.setSlotID(selectedSlot);
                String selSloturl = slot_url.get(i);
                Slot_info.setSloturl(selSloturl);
                Slot_details();
                // TODO get
            }
        });

        Button btnMap = (Button) findViewById(R.id.btnViewMap);
        btnMap.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(View_Parking.this, Map.class);
                startActivity(intent);
            }
        });

        /*Progressdialog dialog = new Progressdialog();
        dg = dialog.createDialog(this); */

    }

    public void Slot_details() {
        final ConnectionM conn = new ConnectionM();
        if (ConnectionM.checkNetworkAvailable(this)) {
            dg1 = new ProgressDialog(this);
            //dg.create();
            dg1.show();

            Thread tthread = new Thread() {
                @Override
                public void run() {
                    try {
                        if (conn.Slot_Info()) {
                            resp = 0;
                        } else {
                            resp = 1;
                        }
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                    hd3.sendEmptyMessage(0);

                }
            };
            tthread.start();
        } else {
            Toast.makeText(getApplicationContext(), "Sorry no network access.",
Toast.LENGTH_LONG).show();
        }
    }

    public Handler hd3 = new Handler() {
        public void handleMessage(Message msg) {
            if(dg1.isShowing())
            {
                dg1.dismiss();
            }
            //dg.cancel();
            switch (resp) {
                case 0:
                    Intent intent = new Intent(View_Parking.this, Slot_Activity.class);
                    startActivity(intent);
                    break;

                case 1:
                    Toast.makeText(getApplicationContext(), "Already Booked",
Toast.LENGTH_LONG).show();
```

```
                break;
            }
        }
    };

    public void fillParkList(final String placeName) {
        final ConnectionM conn = new ConnectionM();
        if (ConnectionM.checkNetworkAvailable(this)) {
            //dg2 = new ProgressDialog(this);
            //dg.create();
            //dg2.show();

            Thread tthread = new Thread() {
                @Override
                public void run() {
                    try {
                        if (conn.FillPark(placeName)) {
                            resp = 0;
                        } else {
                            resp = 1;
                        }
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                    hd2.sendEmptyMessage(0);

                }
            };
            tthread.start();
        } else {
            Toast.makeText(getApplicationContext(), "Sorry no network access.",
  Toast.LENGTH_LONG).show();
        }
    }

    public Handler hd2 = new Handler() {
        public void handleMessage(Message msg) {
            /*if(dg2.isShowing())
            {
                dg2.dismiss();
            }*/
            //dg.cancel();
            switch (resp) {
                case 0:
                    //TODO
                    slot_no = Slots_data.getSlotNo();
                    slot_url = Slots_data.getSlotUrl();
                    slot_id = Slots_data.getSlotId();
                    area_img = Slots_data.getArea_img();

                    Slots_adapter adapter = new Slots_adapter(View_Parking.this, slot_id, slot_no);
                    ListView lst = (ListView) findViewById(R.id.listView);
                    lst.setAdapter(adapter);

//Toast.makeText(getApplicationContext(),ConnectionM.imgUrl+area_img.get(0),Toast.LENGT
H_LONG).show();

Picasso.with(View_Parking.this).load(ConnectionM.imgUrl+area_img.get(0)).into(imgArea);
                    break;

                case 1:
                    Toast.makeText(getApplicationContext(), "data not received",
Toast.LENGTH_LONG).show();

                    break;
            }
        }
    };

    public void Fill_Area() {
        final ConnectionM conn = new ConnectionM();
        if (ConnectionM.checkNetworkAvailable(this)) {
            /*dg3 = new ProgressDialog(this);
            //dg.create();
            dg3.show();*/

            Thread tthread = new Thread() {
                @Override
                public void run() {
                    try {
                        if (conn.fill_arealist()) {
                            resp = 0;
                        } else {
                            resp = 1;
                        }
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                    hd.sendEmptyMessage(0);

                }
            };
            tthread.start();
        } else {
            Toast.makeText(getApplicationContext(), "Sorry no network access.",
    Toast.LENGTH_LONG).show();
        }
    }


    public Handler hd = new Handler() {
        public void handleMessage(Message msg) {
            /*if(dg3.isShowing())
            {
                dg3.dismiss();
            }*/
            //dg.cancel();
            switch (resp) {
                case 0:
                    //TODO
                    ArrayList<String> Names = new ArrayList<String>();
                    Names = Area_data.getList_name();
                    Spinner sp = (Spinner) findViewById(R.id.spinnerarea);
                    sp.setAdapter(new ArrayAdapter<String>(View_Parking.this,
                            android.R.layout.simple_spinner_dropdown_item, Names));

                    lat = Area_data.getLat();
                    lon = Area_data.getLon();
                    //area_img = Area_data.getArea_img();
                    break;

                case 1:
                    Toast.makeText(getApplicationContext(), "data not received",
    Toast.LENGTH_LONG).show();

                    break;
            }
        }
    };

    public void Fill_AreaBy(final String by) {
        final ConnectionM conn = new ConnectionM();
        if (ConnectionM.checkNetworkAvailable(this)) {
            /*dg4 = new ProgressDialog(this);
            //dg.create();
            dg4.show();*/
            Thread tthread = new Thread() {
                @Override
                public void run() {
                    try {
                        if (conn.fill_arealistby(by)) {
                            resp = 0;
                        } else {
                            resp = 1;
                        }
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                    byhd.sendEmptyMessage(0);

                }
            };
            tthread.start();
        } else {
            Toast.makeText(getApplicationContext(), "Sorry no network access.",
Toast.LENGTH_LONG).show();
        }
    }


    public Handler byhd = new Handler() {
        public void handleMessage(Message msg) {
            /*if(dg4.isShowing())
            {
                dg4.dismiss();
            }*/
            //dg.cancel();
            switch (resp) {
                case 0:
                    //TODO
                    ArrayList<String> Names = new ArrayList<String>();
                    Names = Area_data.getList_name();
                    Spinner sp = (Spinner) findViewById(R.id.spinnerarea);
                    sp.setAdapter(new ArrayAdapter<String>(View_Parking.this,
                            android.R.layout.simple_spinner_dropdown_item, Names));

                    lat = Area_data.getLat();
                    lon = Area_data.getLon();

                    break;

                case 1:
                    Toast.makeText(getApplicationContext(), "data not received",
Toast.LENGTH_LONG).show();

                    break;
            }
        }
    };

}
```

- **View QR Code :-**

```
package com.example.cp.qr_car_parking_app;

import android.graphics.Bitmap;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ImageView;

import com.example.cp.qr_car_parking_app.Data.Booking_details;
import com.example.cp.qr_car_parking_app.Data.ImageLoadTask;
import com.google.zxing.BarcodeFormat;
import com.google.zxing.MultiFormatWriter;
import com.google.zxing.WriterException;
import com.google.zxing.common.BitMatrix;
import com.squareup.picasso.Picasso;

public class View_QR extends AppCompatActivity {

   public static int white = 0xFFFFFFFF;
   public static int black = 0xFF000000;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_view__qr);
      ImageView imageView = (ImageView) findViewById(R.id.imageView);
      String str = Booking_details.getBookingId();
      try {
         Bitmap bm = encodeAsBitmap(str);
         imageView.setImageBitmap(bm);
       } catch (Exception e) {
         e.printStackTrace();
      }

      ImageView imgPath=(ImageView)findViewById(R.id.imgBookPath);
      new ImageLoadTask(Booking_details.getImgPath(), imgPath);

      new ImageLoadTask(Booking_details.getImgPath(), imgPath).execute();

   }

   Bitmap encodeAsBitmap(String str) throws WriterException {
      BitMatrix result;
      try {
            result = new MultiFormatWriter().encode(str,
                    BarcodeFormat.QR_CODE, 300, 300, null);
        } catch (IllegalArgumentException iae) {
            // Unsupported format
            return null;
        }
        int w = result.getWidth();
        int h = result.getHeight();
        int[] pixels = new int[w * h];
        for (int y = 0; y < h; y++) {
            int offset = y * w;
            for (int x = 0; x < w; x++) {
                pixels[offset + x] = result.get(x, y) ? black : white;
            }
        }
        Bitmap bitmap = Bitmap.createBitmap(w, h, Bitmap.Config.ARGB_8888);
        bitmap.setPixels(pixels, 0, 300, 0, 0, w, h);
        return bitmap;
    }

}
```

5.2.1 Coding Efficiency

Code efficiency is a broad term used to depict the reliability, speed and programming methodology used in developing codes for an application. Code efficiency is directly linked with algorithmic efficiency and the speed of runtime execution for software. It is the key element in ensuring high performance. The goal of code efficiency is to reduce resource consumption and completion time as much as possible with minimum risk to the business or operating environment. The software product quality can be accessed and evaluated with the help of the efficiency of the code used

- removed unnecessary code or code that goes to redundant processing
- use of optimal memory and non-volatile storage
- ensured the best speed or run time for completing the algorithm
- use of reusable components wherever possible
- have use of error and exception handling at all layers of software, such as the user interface, logic and data flow
- created programming code that ensures data integrity and consistency
- developed programming code that's compliant with the design logic and flow
- have use of coding practices applicable to the related software
- optimized the use of data access and data management practices
- used best keywords, data types and variables, and other available programming concepts to implement the related algorithm.

5.3 TESTING APPROACH

5.3.1 Unit Testing:

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. It forms the basis for component testing.

Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

5.3.2 Integration Testing:

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing.

Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. These "design items", i.e., assemblages (or groups of units), are exercised through their interfaces using black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test whether all the components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e., unit testing. The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages.

Software Integration Testing is performed according to the Software Development Life Cycle (SDLC) after module and functional tests. The cross-dependencies for software integration testing are: schedule for integration testing, strategy and selection of the tools used for integration, define the cyclomatic complexity of the software and software architecture, reusability of modules and life-cycle / versioning management.

Some different types of integration testing are big bang, top-down, and bottom-up, mixed (sandwich) and risky - hardest. Other Integration Patterns are: Collaboration Integration, Backbone Integration, Layer Integration, Client/Server Integration, Distributed Services Integration and High-frequency Integration.

5.3.3 System Testing:

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic

As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

5.3.4 System Integration Testing:

System integration testing involves the overall testing of a complete system of many subsystem components or elements. The system under test may be composed of hardware, or software, or hardware with embedded software, or hardware/software with human-in-the-loop testing.

SIT consists, initially, of the "process of assembling the constituent parts of a system in a logical, cost-effective way, comprehensively checking system execution (all nominal & exceptional paths), and including a full functional check-out." Following integration, system test is a process of "verifying that the system meets its requirements, and validating that the system performs in accordance with the customer or user expectations."

In technology product development, the beginning of system integration testing is often the first time that an entire system has been assembled such that it can be tested as a whole. In order to make system testing most productive, the many constituent assemblies and subsystems will have typically gone through a subsystem test and successfully verified that each subsystem meets its requirements at the subsystem interface level.

In the context of software systems and software engineering, system integration testing is a testing process that exercises a software system's coexistence with others. With multiple integrated systems, assuming that each have already passed system testing, SIT proceeds to test their required interactions. Following this, the deliverables are passed on to acceptance testing.

Testing Strategies

The basic strategies that were used for testing were following

- Specification Testing
- Black Box Testing
- White Box Testing
- Regression Testing
- Acceptance Testing
- Assertion Testing
- Unit Testing

- **Specification Testing**

Even if the code testing is performed exclusively, it doesn’t ensure against program failure. Code testing doesn’t answer whether the code meets the agreed specifications document. It doesn’t also determine whether all aspects of the design are implemented. Therefore, examining specifications stating what program should do and how it should behave under various conditions performs specification testing.

- **Black Box Testing**

Black-box testing is conducted at the software interface. In Black Box testing only the functionality was tested without any regard to the code written. If the functionality, which was expected from a component, is provided then black box testing is completed.

- **White Box Testing**

White-box testing, sometimes called glass-box testing is a test case design method that uses the control structure of the procedural design to derive test cases. In White Box testing internal code written in every component wasted and it was checked that the code written is efficient in utilizing various resources of the system like memory or the utilizing of input output

- **Regression Testing**

In Regression testing the software was tested against the boundary conditions. Various input fields were tested against abnormal values and it was tested that the software does not behave abnormally at any time.

- **Acceptance Testing**

In acceptance testing the software was checked for completeness that itis ready. Normally the quality assurance department performs the acceptance testing that the software is ready and can be exported.

5.3.5 Beta Testing

Beta testing also known as user testing takes place at the end users' site by the end users to validate the usability, functionality, compatibility, and reliability testing. Beta testing adds value to the software development life cycle as it allows the "real" customer an opportunity to provide inputs into the design, functionality, and usability of a product. These inputs are not only critical to the success of the product but also an investment into future products when the gathered data is managed effectively.

5.4 MODIFICATION AND IMPROVEMENTS

Future improvements

The improvement is the only way to ahead in the future. However, our application may seem perfect, but it might not be and has much scope of improvement.

- The next version of the software will be more updated version.
- Easy placements of controls & navigations cues allow further modification into user interface & addition of controls very easily.
- It will provide user with more high performance & more stability than this.
- Logical & technical errors will be less found in future.
- System will be more user friendly as compared to today's system.
- It will be easy to locate the necessary information.
- Provide feedback module that has wide number of subjects for all type of users.
- The application can be enhanced in the near future.
- Almost totally paperless system can be provided.

Security has to be improved by implementing firewall & other suitable system

5.5 <u>SYSTEM MAINTENANCE AND EVALUTION</u>

5.5.1 MAINTENANCE

Maintenance is an enigma of the system development. It holds the software industry captive. Analysts spend more time in maintaining programs than coding them. Software maintenance denotes any changes made to the software product after it has been delivered to the customer. Most products need maintenance due to the wear and tear of the product. Software Maintenance can be divided into following types:

- **Corrective Maintenance:** It is necessary to rectify the bugs observed while the system is in use. **Smart Parking System** needs this maintenance for any removing flaws that can arise while sending the data or for correcting the logical bugs that might have been left unchecked as they appear only in real time like empty database.
- **Perfective Maintenance:** Software product might need maintenance to support the new features that users want it to support, to change different functionalities of the system according to the customer demands, or to enhance the performance of the system. **Smart Parking System** needs this maintenance for removing the short falls of its current features.

Software Maintenance is essential as initial stages of any software developed are always unstable. Over the time it achieves stability as bugs are fixed and faults are removed to make the system accurate. System Maintenance is often termed as the task of doing repairs to the developed system. When websites are inaccessible due to attacks from hackers, server problems or for updating and repair, the administrators of the website will often display an image apologizing for System Maintenance and Website downtime. This allows the user to understand that the website cannot be used and that the administrators are aware of the issue.

5.5.2 EVALUATION

System Evaluation is termed as the task of evaluating the success and failure of the system. It is performed with the help of following two V ‘s:

- **Verification:**

Verification determines whether the system is built correctly and does not contain technical errors. It also involves the review of the requirements, to verify that the right problem is being solved. Verification also ensures that the system is syntactically and logically correct and performs functionally as being specified. It is a static practice of verifying documents, design, code and program.

As verification relates to the humanized effort of checking the documents and files, we have taken utmost care to see to it that the application conforms to specifications. Reviews and inspections were carried out periodically. The web-based application has been put through the process of Verification successfully.

- **Validation:**

Validation on the other hand is a difficult task of ensuring the meaning and content of the rules meet some carefully defined criteria of adequacy. Defining such criteria is the key to successfully conduct Validation procedure and demonstrating the level of acceptability of the system.

As Validation is a dynamic mechanism of validating and testing the actual product; we have implemented the process of validation by executing the code thoroughly. By performing White Box as well as Black Box testing; along with Acceptance Testing, we have made sure that the application adheres to customer ‘s expectation and requirements.

The target for validation was actual product-a unit, a module, a bent of integrated modules, and effective final product.

- **Verification** process describes whether the outputs are according to inputs or not.

- **Validation** process describes whether the software is accepted by the user or not.

5.6 TEST CASE TABLE

Test Case Id	Test Case	Expected Result	Actual Result	Result
1	User Registration	User should be able to register himself His mobile no, email id should be validated.	User was able to register himself. All client-side validation was taken care.	Pass
2	Login with correct user name and password	User should be successfully login to account.	User was able to login successfully	Pass
3	User Login with incorrect username and password	User should be not allowed to login and application must display "Incorrect Login Details"	Application displayed "Incorrect Login Details" and user was not allow to login	Pass
4	Login with correct admin name and password	Admin should be successfully login to account.	Admin was able to login successfully	Pass
5	Admin Login with incorrect username and password	Admin should be not allowed to login and error massage shows	Admin was not able to login.	Pass
6	Add User with valid information	User should add successfully	User was added successfully	Pass

7	Login with correct user name and password	App should be able to login successfully.	App was denied the login.	Fail
8	Registration for Parking system	App should be able to register successfully.	App was denied the registration.	Fail
9	Forgot password for parking system	App will be able to a new password	App was able to set a new password.	Pass
10	Registration for Parking System	App should be able to register successfully.	App was able to register.	Pass
11	Login with correct admin name and password	App should be able to login successfully.	App was able to login.	Pass

Table 6: Test Case Table

CHAPTER 6
RESULTS AND DISCUSSION

6.1 TEST REPORT

Cross browser testing and mobile testing was done by the us to ensure that the web application looks the same in major browsers that is Google chrome, Mozilla Firefox and Internet explorer and on different version of android phone. The web-based application project is consistent (looks exactly the same) in Google chrome and Mozilla Firefox but the looks vary slightly in internet explorer.

6.2 USER DOCUMENTATION

- **WEBSITE INTERFACE**

1. ADMIN PAGE

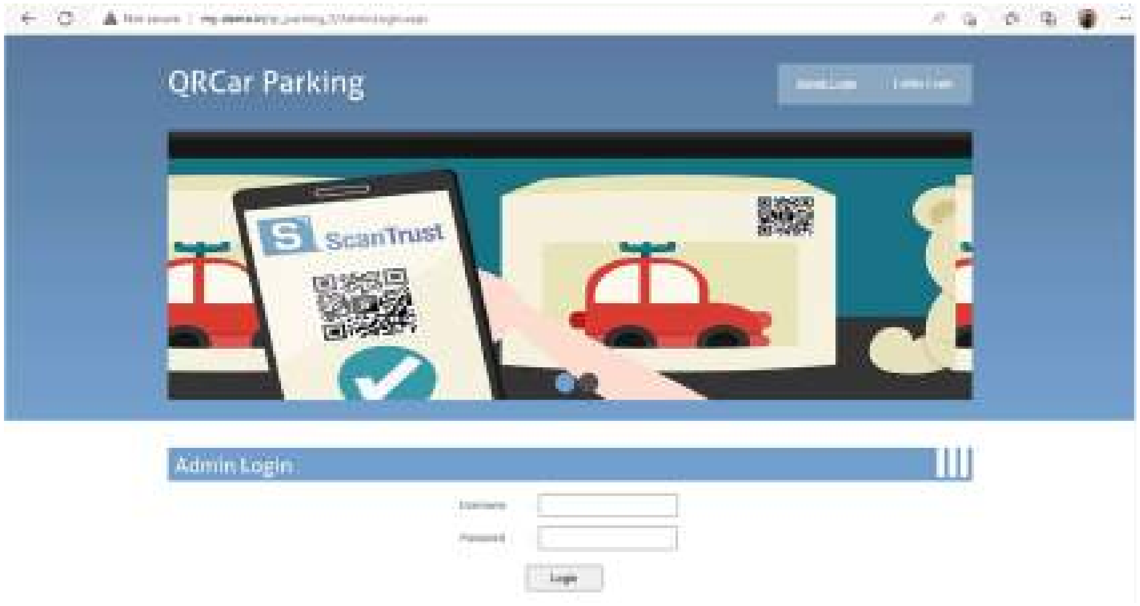


Figure 49: Admin Page

This is the Log in page for admin where he has to enter his username and password.

2. HOME PAGE

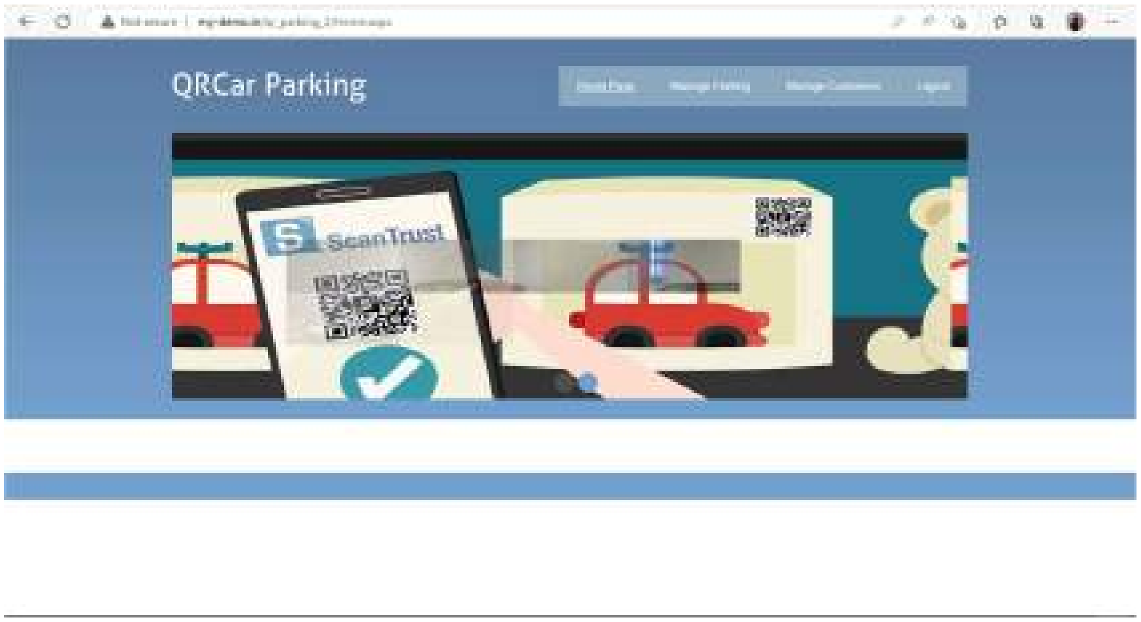
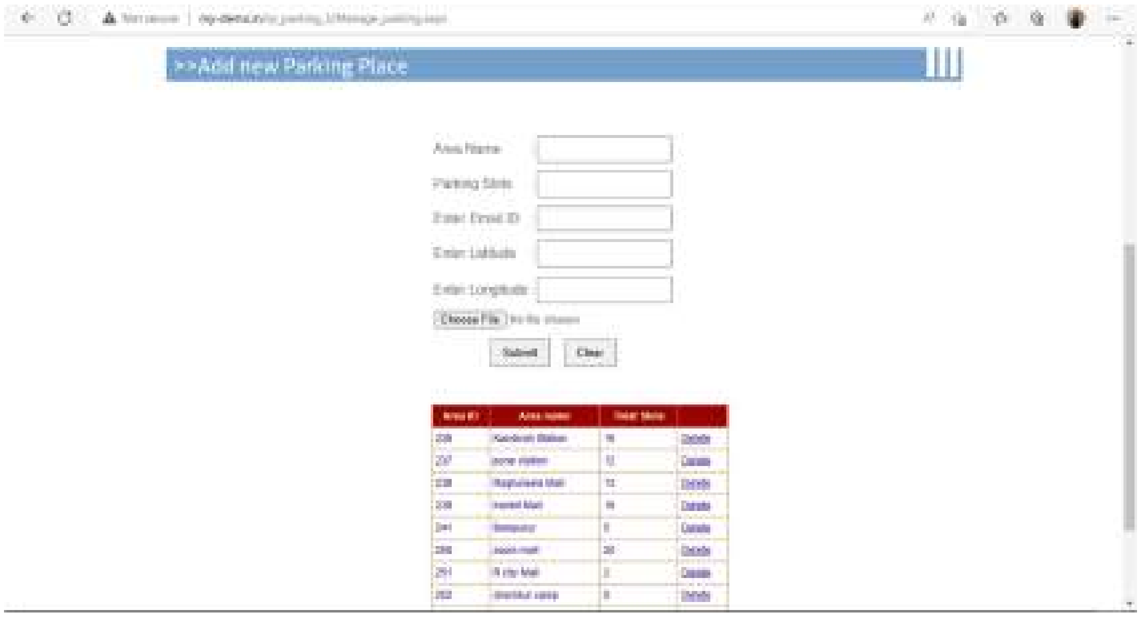


Figure 50: Home Page

This the Home page for admin where he can manage parking ,add parking spaces, manage customer as shown in figure 50

3. ADDING NEW PARKING PLACE


>>Add new Parking Place

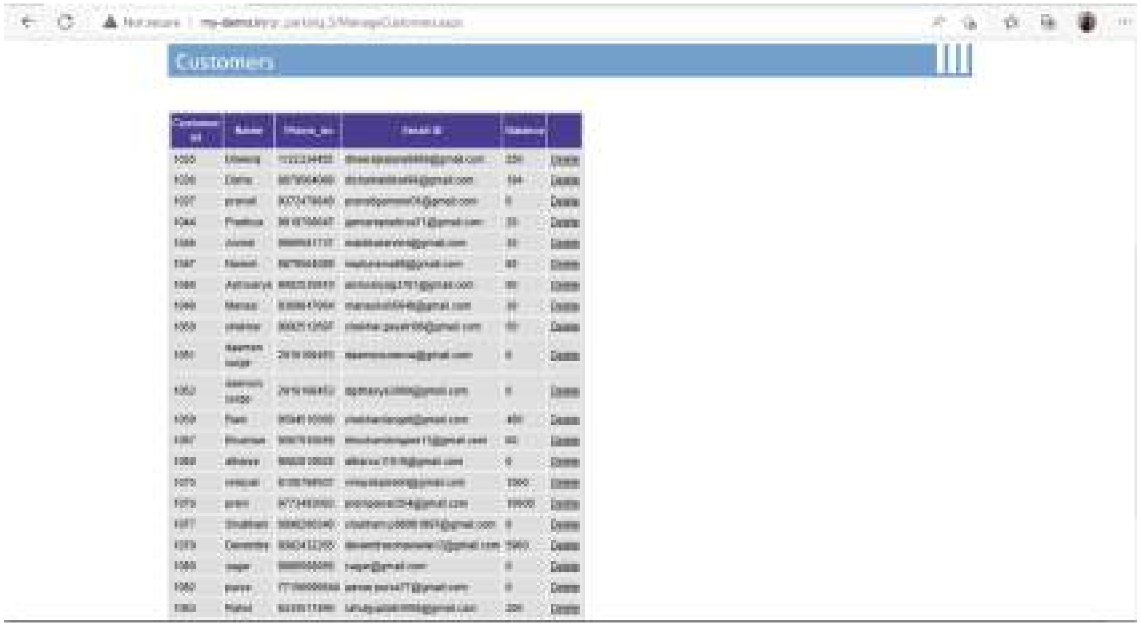
Area Name
Parking Slots
Enter Email ID
Enter Latitude
Enter Longitude

Area ID	Area name	Total Slots	
236	Kandivali Station	15	Details
237	[illegible] station	12	Details
238	Raghuleela Mall	15	Details
239	Inorbit Mall	15	Details
241	[illegible]	0	Details
250	[illegible] mall	20	Details
251	R city Mall	2	Details
252	[illegible]	0	Details

Figure 51: Adding New Parking Place

After clicking on "Add new Parking Place" the admin can add new parking spaces by entering the appropriate details into the fields as shown in the Figure 51

4. MANAGE CUSTOMER


Customers

Customer Id	Name	Phone_no	Email ID	Distance	
[illegible]	[illegible]	[illegible]	[illegible]@gmail.com	230	Delete
[illegible]	Dishu	[illegible]	[illegible]@gmail.com	194	Delete
[illegible]	pranali	[illegible]	[illegible]@gmail.com	0	Delete
[illegible]	Pratiksa	[illegible]	[illegible]@gmail.com	20	Delete
[illegible]	[illegible]	[illegible]	[illegible]@gmail.com	30	Delete
[illegible]	Naresh	[illegible]	[illegible]@gmail.com	60	Delete
[illegible]	Aishwarya	[illegible]	[illegible]@gmail.com	90	Delete
[illegible]	Manasi	[illegible]	[illegible]@gmail.com	30	Delete
[illegible]	shekhar	[illegible]	[illegible]@gmail.com	50	Delete
[illegible]	[illegible]	[illegible]	[illegible]@gmail.com	0	Delete
[illegible]	[illegible]	[illegible]	[illegible]@gmail.com	0	Delete
[illegible]	Ravi	[illegible]	[illegible]@gmail.com	400	Delete
[illegible]	Bhushan	[illegible]	[illegible]@gmail.com	60	Delete
[illegible]	[illegible]	[illegible]	[illegible]@gmail.com	0	Delete
[illegible]	vinayak	[illegible]	[illegible]@gmail.com	1000	Delete
[illegible]	prem	[illegible]	[illegible]@gmail.com	10000	Delete
[illegible]	Shubham	[illegible]	[illegible]@gmail.com	0	Delete
[illegible]	Devendra	[illegible]	[illegible]@gmail.com	5000	Delete
[illegible]	sagar	[illegible]	sagar@gmail.com	0	Delete
[illegible]	purva	[illegible]	[illegible]@gmail.com	0	Delete
[illegible]	Rahul	[illegible]	[illegible]@gmail.com	200	Delete

Figure 52: Manage Customer

5. LOBBY LOGIN

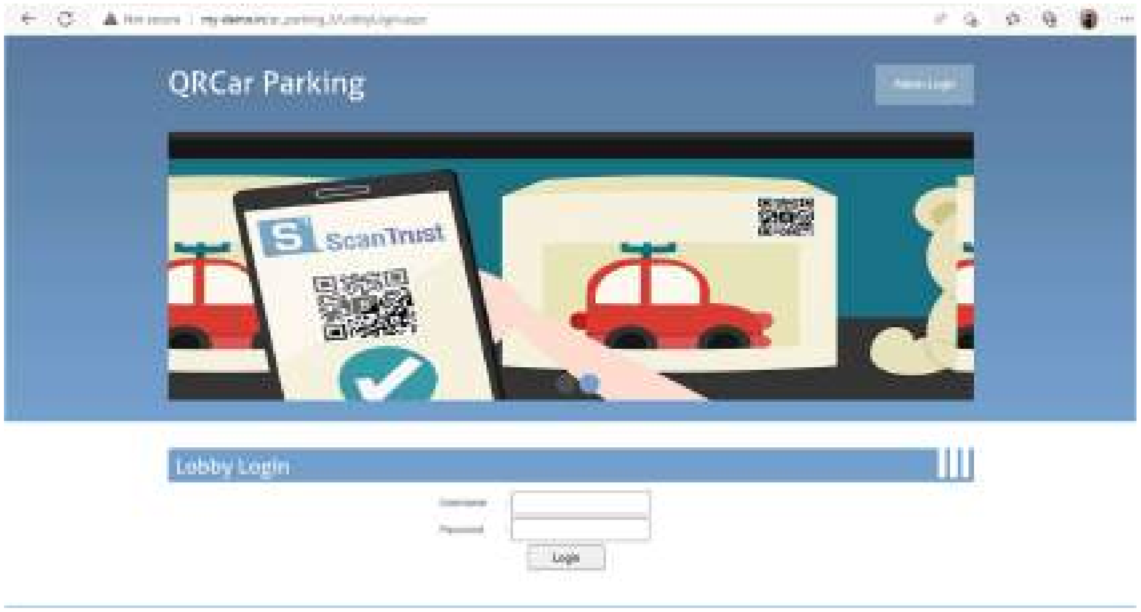
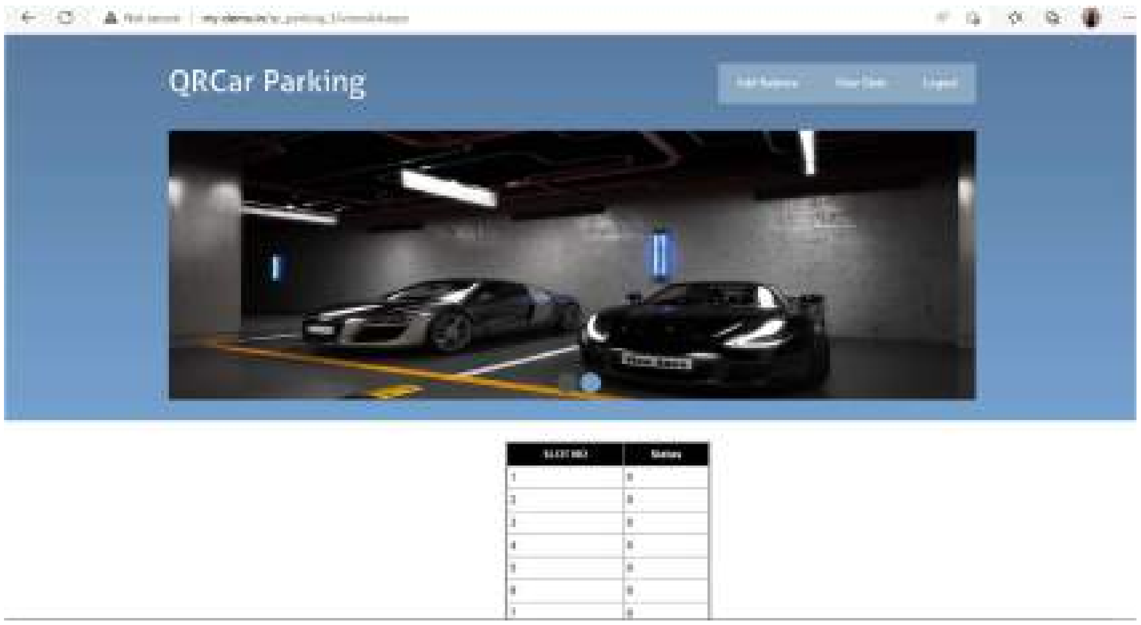


Figure 53: Lobby Login

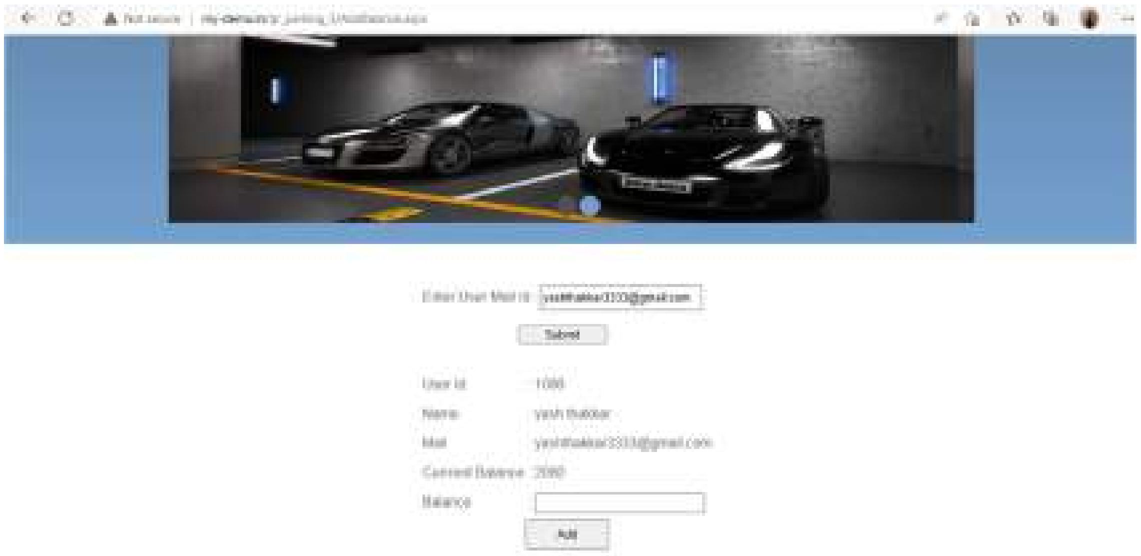
6. VIEW SLOT



SLOT NO	Status
1	0
2	0
3	0
4	0
5	0
6	0
7	0

Figure 54: View Slot

7. ADD BALANCE



Not secure

Enter User Mail Id yashthakkar3333@gmail.com

Submit

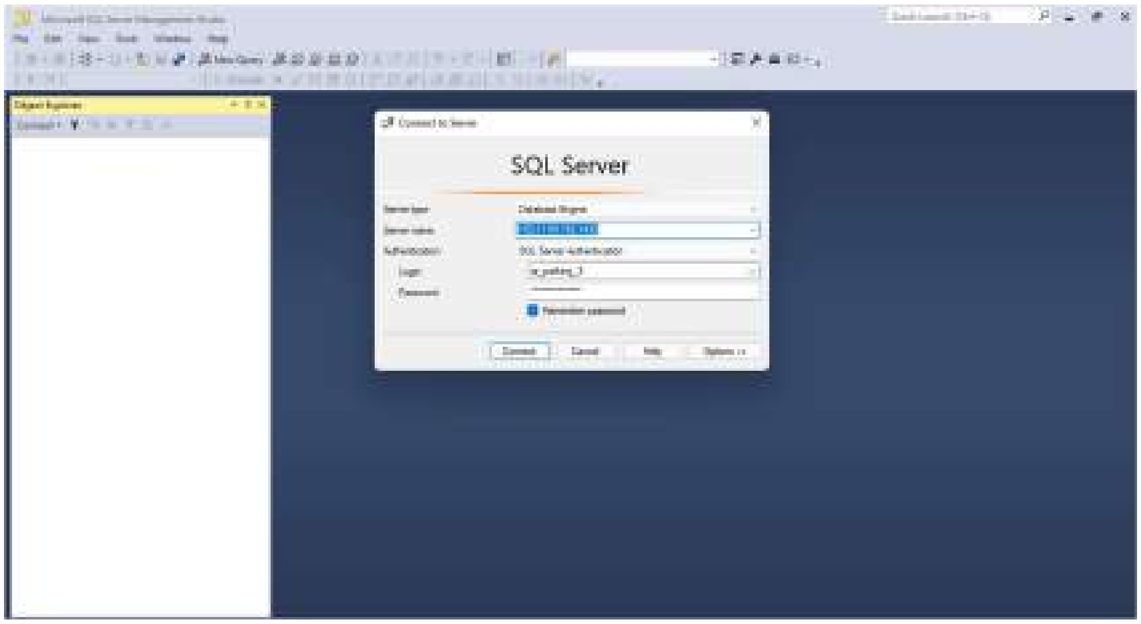
User Id	1000
Name	yash thakkar
Mail	yashthakkar3333@gmail.com
Current Balance	2000
Balance	

Add

Figure 55: Add Balance

- <u>**DATA BASE INTERFACE**</u>

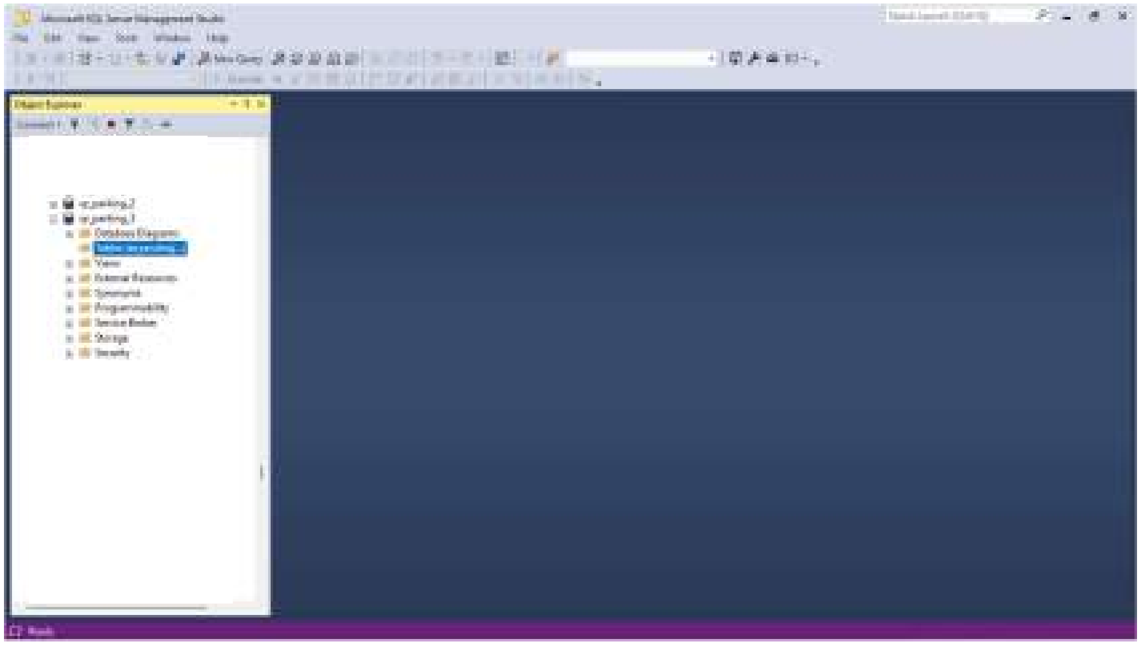
1. SERVER CONNECTION

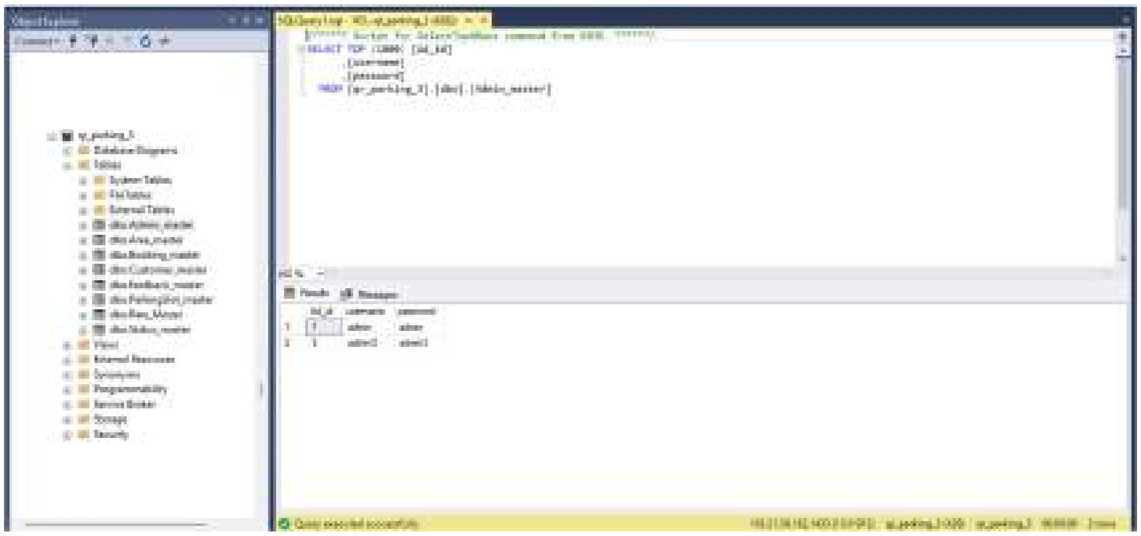


Server type: Database Engine
Server name: [illegible]
Authentication: SQL Server Authentication
Login: [illegible]
Password: [illegible]
[✓] Remember password

Figure 56: Server Connection

2. DIFFERENT DATA

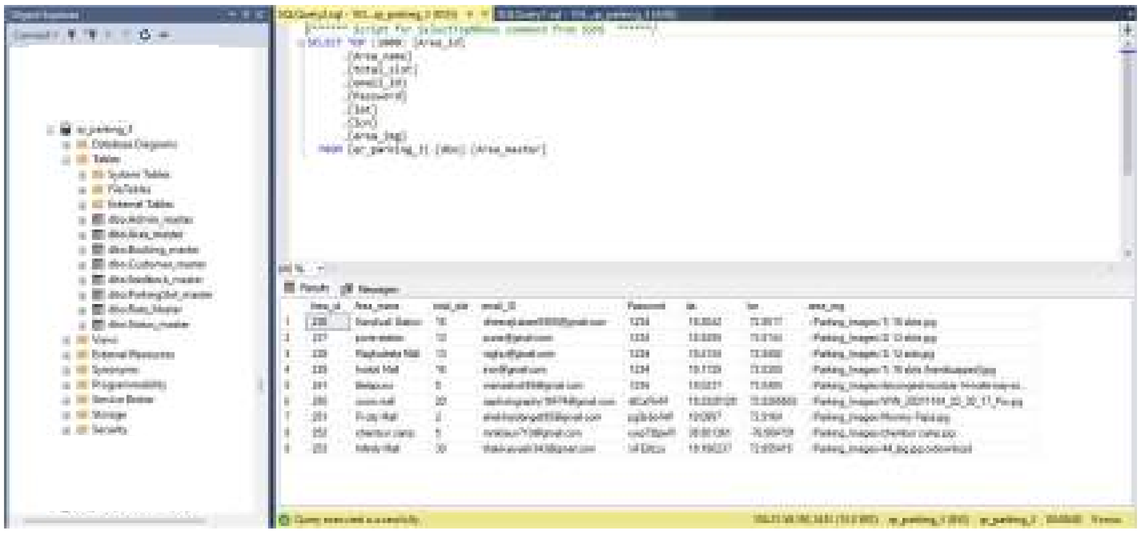




```
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP (1000) [Ad_Id]
      ,[username]
      ,[password]
  FROM [qr_parking_3].[dbo].[Admin_master]
```

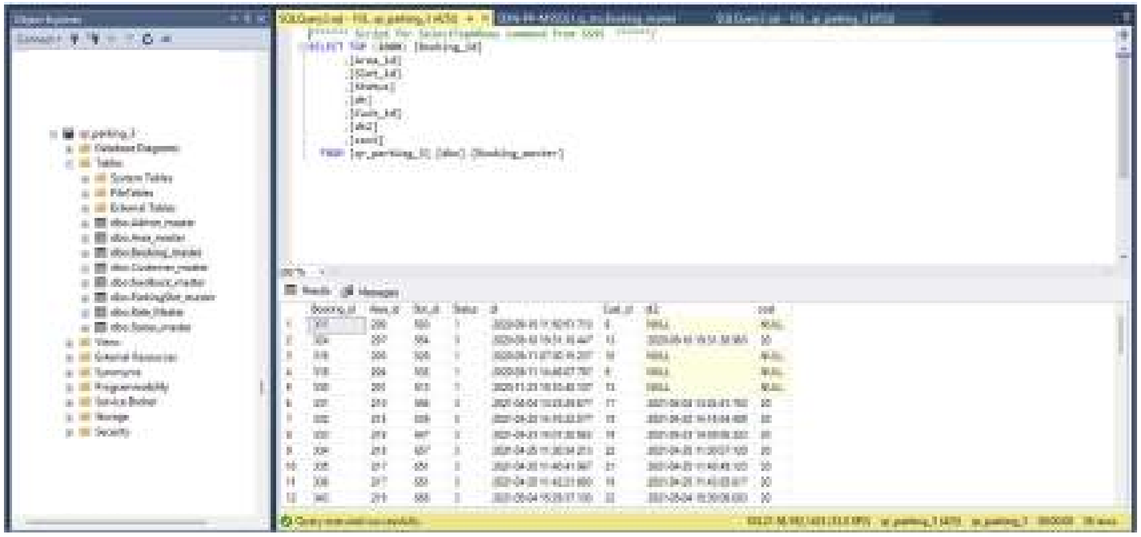
```
   Ad_id  username  password
1  1      admin     admin
2  1      admin2    admin1
```

Query executed successfully.



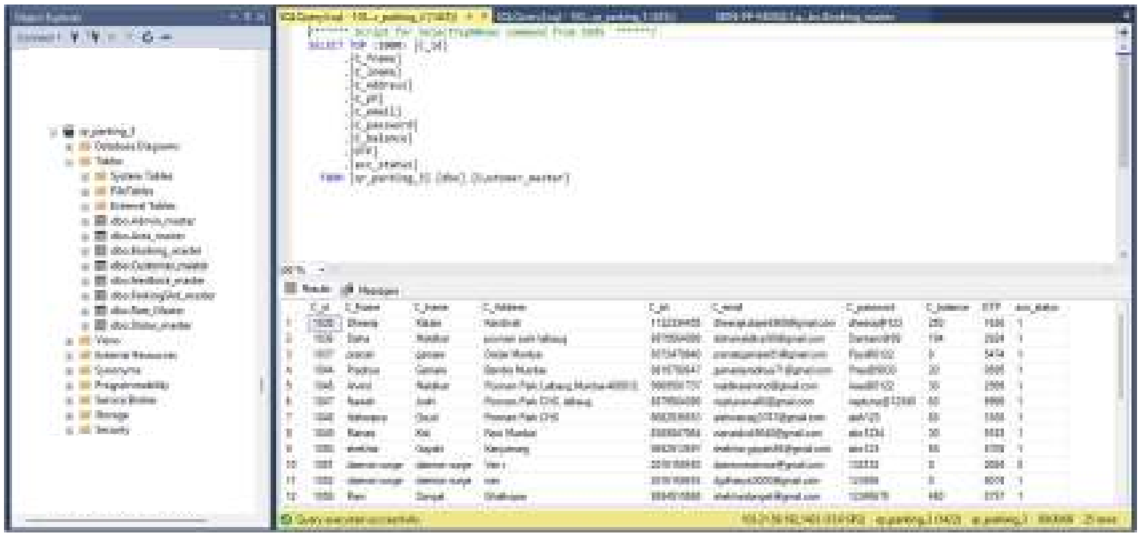
```
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP (1000) [Area_id]
      ,[Area_name]
      ,[total_slot]
      ,[email_ID]
      ,[Password]
      ,[lat]
      ,[lon]
      ,[area_img]
  FROM [qr_parking_1].[dbo].[Area_master]
```

	Area_id	Area_name	total_slot	email_ID	Password	lat	lon	area_img
1	236	Nandivali Station	16	[illegible]@gmail.com	1234	19.0042	73.0617	~Parking_images/1. 16 slots.jpg
2	237	pune station	12	[illegible]@gmail.com	1234	19.0209	73.0744	~Parking_images/2. 12 slots.jpg
3	238	Raghuleela Mall	12	[illegible]@gmail.com	1234	19.0739	73.0002	~Parking_images/3. 12 slots.jpg
4	239	Inorbit Mall	16	[illegible]@gmail.com	1234	19.1739	73.0200	~Parking_images/1. 16 slots [illegible].jpg
5	241	[illegible]	5	[illegible]@gmail.com	1234	19.0217	73.0495	~Parking_images/[illegible]
6	250	[illegible]	20	[illegible]@gmail.com	[illegible]	19.0320129	73.0200003	~Parking_images/WIN_20211104_20_30_17_Pro.jpg
7	251	[illegible] Mall	2	[illegible]@gmail.com	[illegible]	19.0997	72.9164	~Parking_images/[illegible].jpg
8	252	[illegible]	5	[illegible]@gmail.com	[illegible]	20.001264	73.968478	~Parking_images/[illegible].jpg
9	253	[illegible] Mall	30	[illegible]@gmail.com	[illegible]	19.156227	72.935415	~Parking_images/44_jpg.jpg [illegible]



```
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP (1000) [Booking_id]
      ,[Area_id]
      ,[Slot_id]
      ,[Status]
      ,[dt]
      ,[Cust_id]
      ,[dt2]
      ,[cost]
  FROM [qr_parking_1].[dbo].[Booking_master]
```

	Booking_id	Area_id	Slot_id	Status	dt	Cust_id	dt2	cost
1	[illegible]	236	503	1	[illegible]	6	NULL	NULL
2	[illegible]	237	554	1	[illegible]	13	[illegible]	30
3	[illegible]	236	505	1	[illegible]	16	NULL	NULL
4	[illegible]	236	505	1	[illegible]	6	NULL	NULL
5	[illegible]	236	613	1	[illegible]	13	NULL	NULL
6	[illegible]	213	566	3	[illegible]	17	[illegible]	30
7	[illegible]	213	539	3	[illegible]	19	[illegible]	30
8	[illegible]	213	647	3	[illegible]	19	[illegible]	20
9	[illegible]	213	657	3	[illegible]	22	[illegible]	20
10	[illegible]	217	651	3	[illegible]	21	[illegible]	30
11	[illegible]	217	651	3	[illegible]	19	[illegible]	30
12	[illegible]	219	658	3	[illegible]	22	[illegible]	30



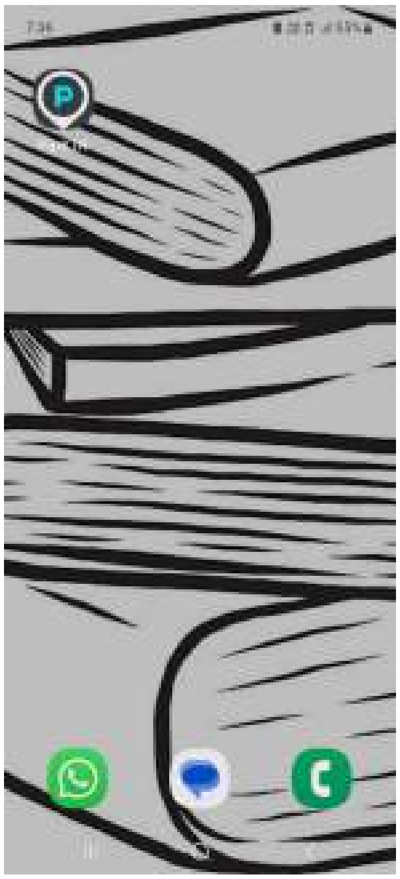
```
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP (1000) [C_id]
      ,[C_fname]
      ,[C_lname]
      ,[C_address]
      ,[C_ph]
      ,[C_email]
      ,[C_password]
      ,[C_balance]
      ,[OTP]
      ,[acc_status]
  FROM [qr_parking_1].[dbo].[Customer_master]
```

	C_id	C_fname	C_lname	C_Address	C_ph	C_email	C_password	C_balance	OTP	acc_status
1	[illegible]	Dheeraj	Kadam	[illegible]	[illegible]	[illegible]@gmail.com	[illegible]	250	[illegible]	1
2	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]@gmail.com	[illegible]	194	[illegible]	1
3	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]@gmail.com	[illegible]	0	5474	1
4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]@gmail.com	[illegible]	20	[illegible]	1
5	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]@gmail.com	[illegible]	30	[illegible]	1
6	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]@gmail.com	[illegible]	50	[illegible]	1
7	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]@gmail.com	[illegible]	50	[illegible]	1
8	[illegible]	[illegible]	[illegible]	Navi Mumbai	[illegible]	[illegible]@gmail.com	[illegible]	30	[illegible]	1
9	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]@gmail.com	[illegible]	65	[illegible]	1
10	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]@gmail.com	[illegible]	0	[illegible]	0
11	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]@gmail.com	[illegible]	0	[illegible]	1
12	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]@gmail.com	[illegible]	440	[illegible]	1

Figure 57: Different Data

- **<u>ANDROID INTERFACE</u>**

1. APP LOGO



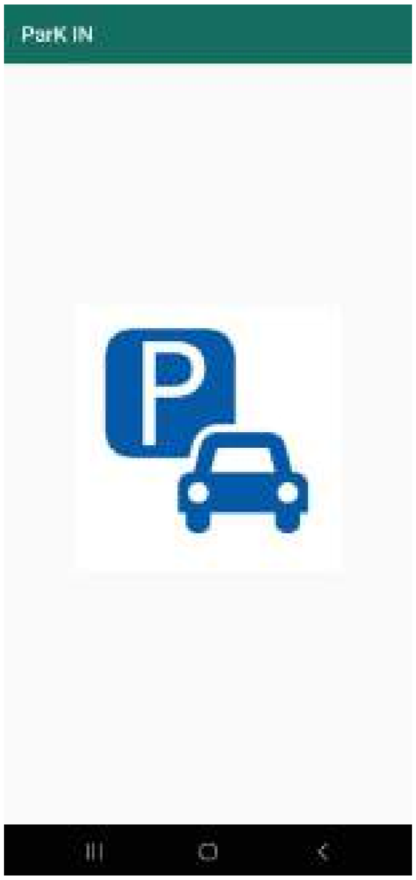


Figure 58: App Logo

Menu of a smart phone showing the smart phone application short cut icon marked

2. HOME PAGE

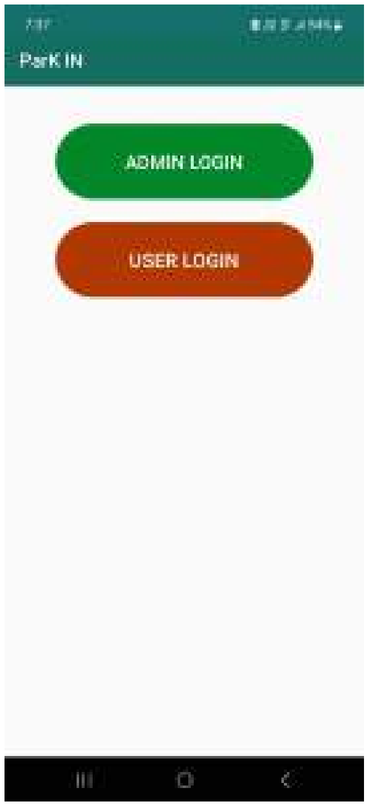


Figure 59: Home Page

3. ADMIN PAGE

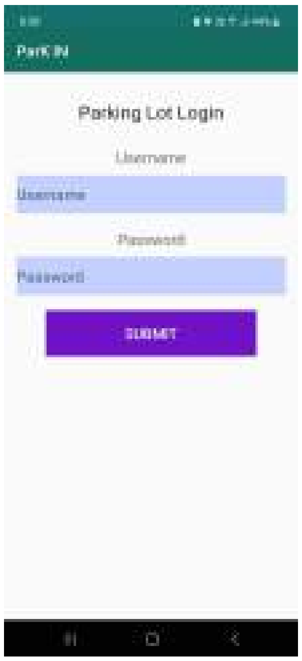


Figure 60: Admin Page

4. **QR CODE SCANNER**

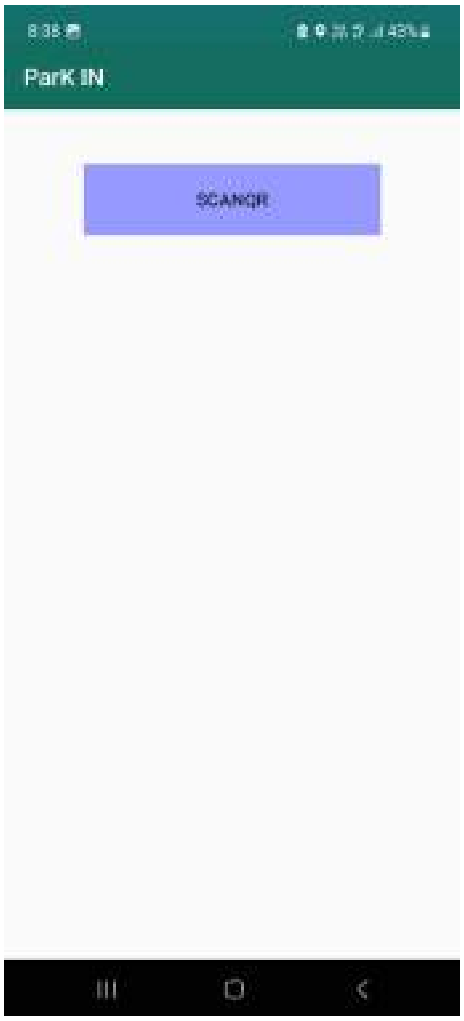


Figure 61: QR Code Scanner

This is a QR code scanner where the admin has to scan the QR code generated by the user when they reaches the entry point .

5. USER LOGIN

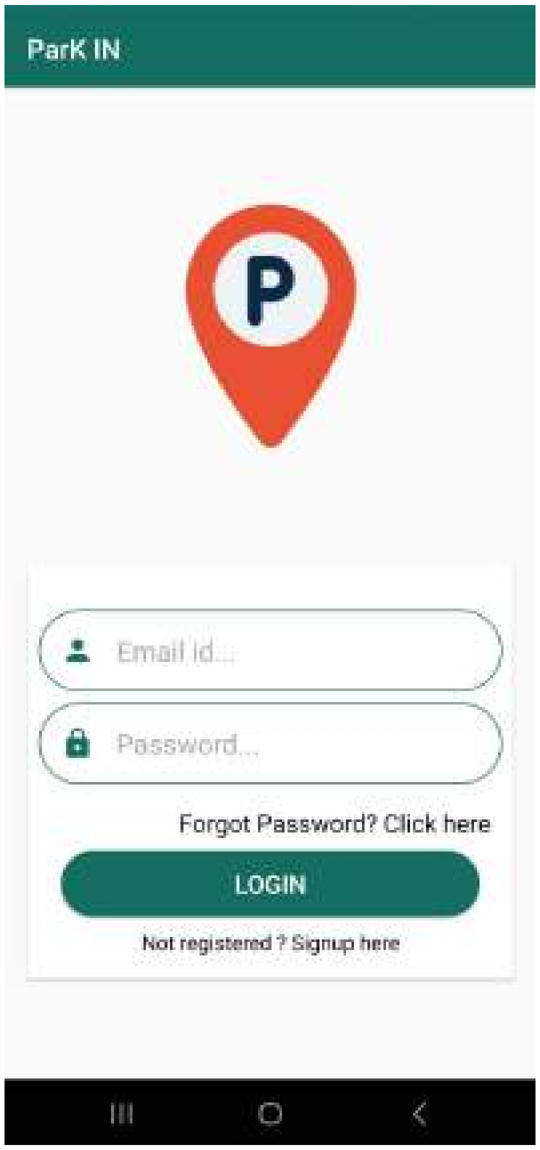


Figure 62: User Login

6. USER REGISTRATION

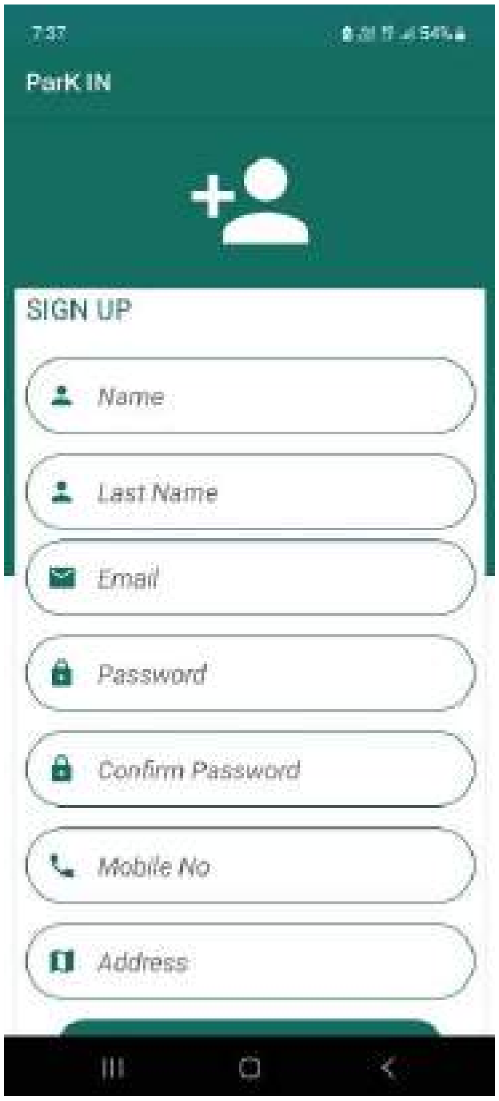


Figure 63: User Registration

After clicking on registration button, the user will have to register himself by entering the appropriate details into the fields asshown in the Figure 7.3

7. USER DASHBOARD

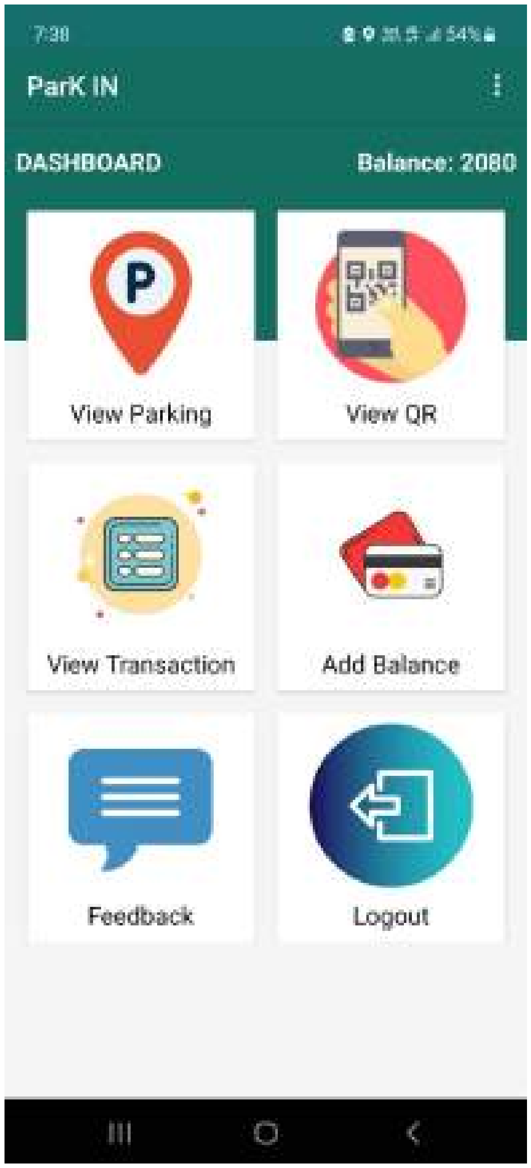
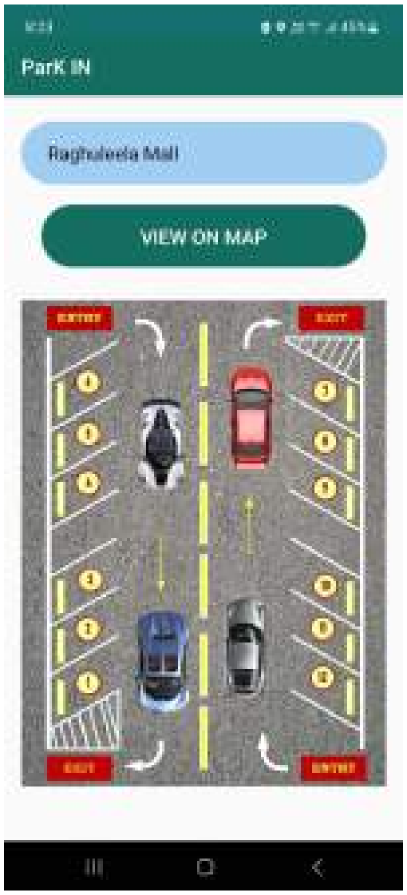


Figure 64: User Dashboard

This is the User Dashboard where he can do the main activity from booking to viewing transactions as shown in figure 64

8. SLOT BOOKING AND MAP



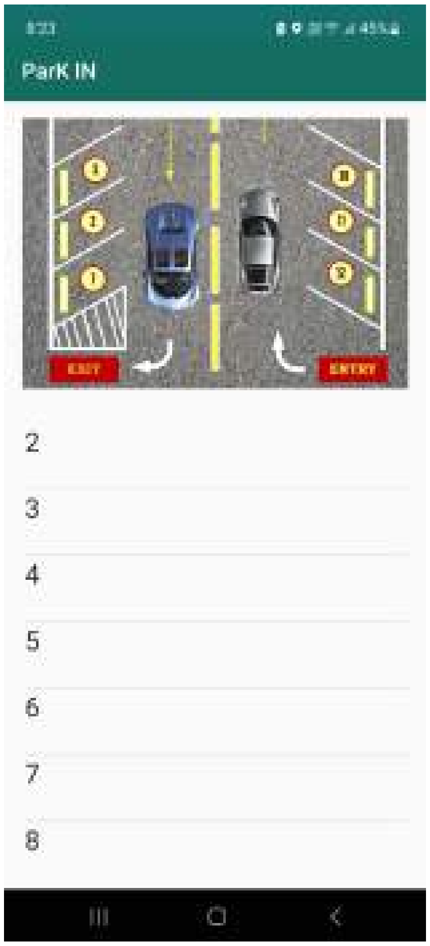


Figure 65: Slot Booking and Map

The user is provided with multiple parking locations. User has to select one of the locations provided where he desires to park the vehicle. After selecting the location, the remaining details including parking time(in time and out time),vehicle number, card expiry date, card holder name ,card number, and then vcc/cvv number and then the user has to do the payment and it is shown below in fig 63 & 65hh

9. ADDING BALANCE

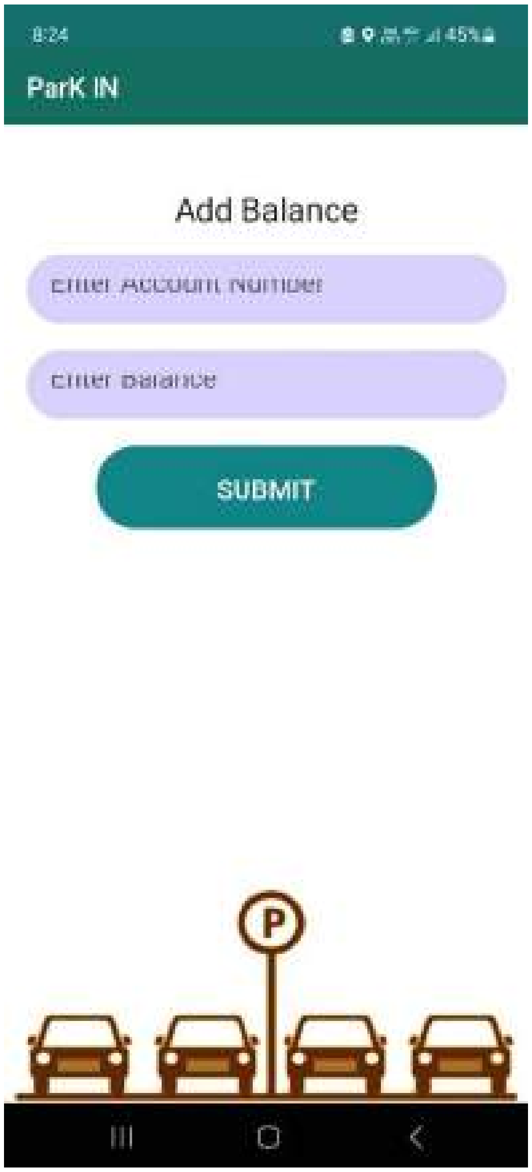
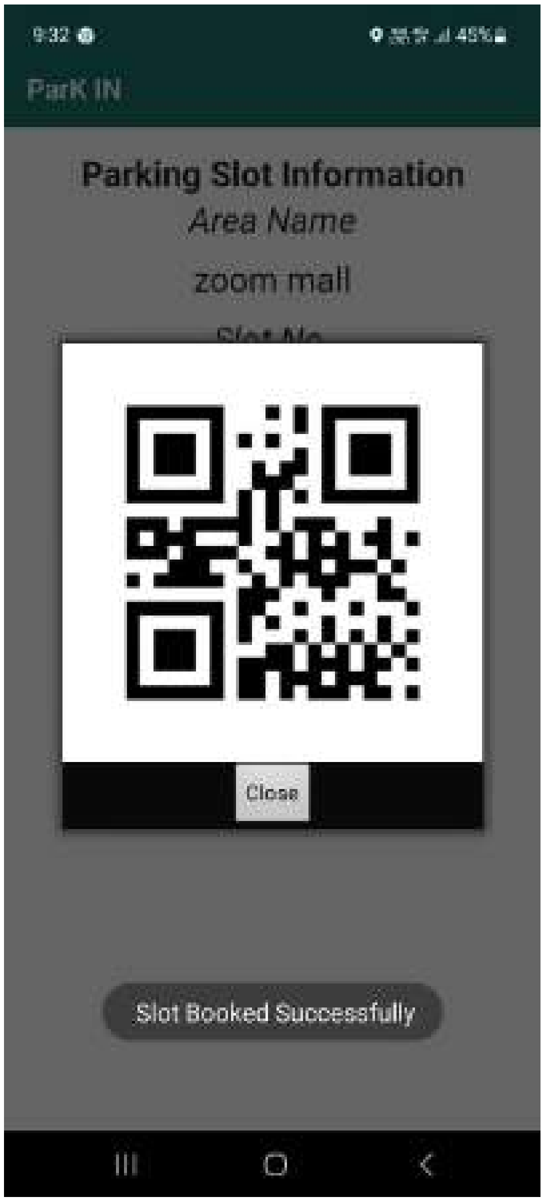


Figure 66: Adding Balance

The user can add money in the form of wallet which can be used while exiting the parking space.

10. QR CODE



Parking Slot Information

Area Name

zoom mall

Slot No

Slot Booked Successfully

Figure 67: QR Code

After booking a slot user will get a barcode which he has to scan when entering the parking space .

11. FEEDBACK

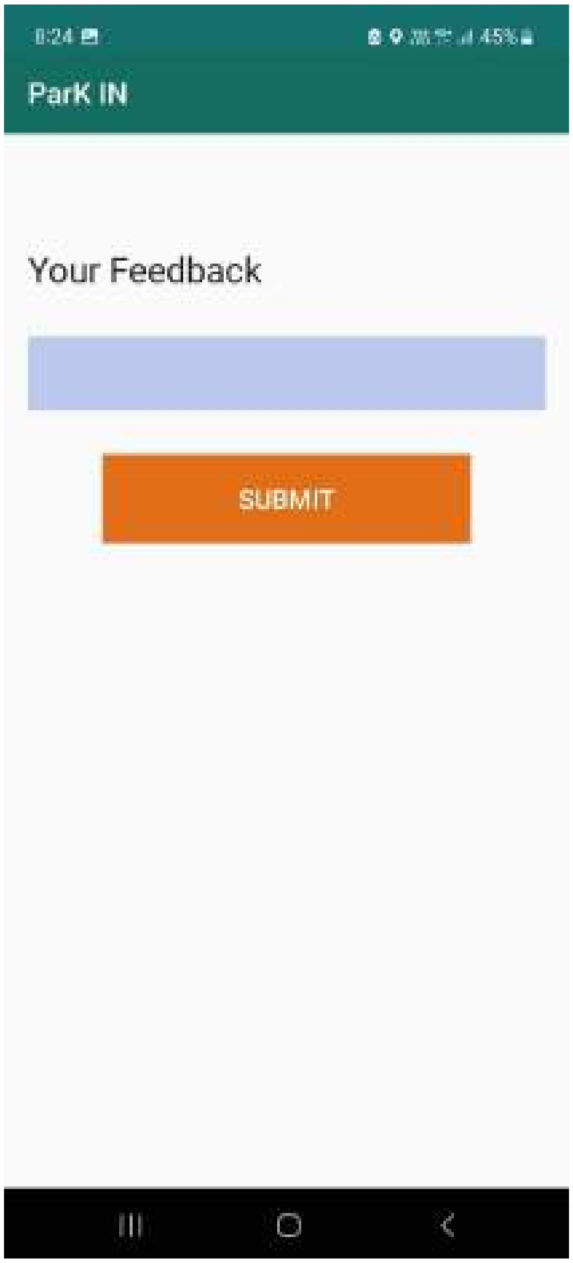


Figure 68: Feedback

CHAPTER 7

CONCLUSION

7.1 <u>CONCLUSION</u>

QR-Code technology would be more easily integrated into existing parking system infrastructures. QR-Code provides all the features which make it a valid technology for mass parking locations like malls, college campus etc.: contactless transactions at high speed, stability and simplicity. The proposed solutions based on combinations of standards and technologies using current contactless infrastructures. Our proposed application will be feasible for novice users as well as professional users. The proposed application will be used for the booking a parking slot without waiting in long queues during peak hours. This android application reduces the manual work of both user as well as parking staff. It is basically the transition from a manual to digital system for parking slot booking. Thus, the problem associated with finding parking slot is almost solved.

7.1.1 SIGNIFICANCE OF THE SYSTEM

This project would be mainly focused on assisting driver to easily find vacant parking spaces in a specific parking region with the help of QR code based Smart Parking System and to reduce traffic and energy consumption and air pollution. Thus, this project has come up with an optimal solution that gives liberty to the people to book their own parking space as per their need and specification of the vehicle. The purpose of this project is to make people more convenient to park their vehicle, which in this case is Reservation Based Smart Parking System, the question to be addressed here in this module is, how to give parking slots to the drivers? The project is to mainly answer this particular question addressed by providing an Android application to reserve parking slot as per drivers need.

7.2 LIMITATIONS OF THE SYSTEM

Although I have put my best efforts to make the software flexible, easy to operate but limitations cannot be ruled out even by me. Though the software presents a broad range of options to its users some intricate options could not be covered into it; partly because of logistic and partly due to lack of sophistication. Lack of time was also major constraint; thus, it was not possible to make the software foolproof and dynamic. Lack of time also compelled me to ignore some part such as storing old result of the candidate etc.

Considerable efforts have made the software easy to operate even for the people not related to the field of computers but it is acknowledged that a layman may find it a bit problematic at the first instance. The user is provided help at each step for his convenience in working with the software.

- Android Smartphone is must.
- Active Internet Connection is required.
- User cannot cancel his booking.
- Since this project is for the final project the credit card is not working properly

7.3 FUTURE ENHANCEMENT

This project has a wide scope for future development, as the user‘s requirement is always going to be changed which is not static and these needs are dynamic. The technology which is famous today becomes outdated in very next day. To keep abstract of technical improvements, the system may be further refined. So, such type of system is improved in further future development. This enhancement is done in an efficient and effective manner. We can thus update the same with further modification establishment and can be integrated with minimal modification. Thus, the project is extendable and can be developed in anytime with more advanced features.

Some advance future enhancements:

- This proposed system can be integrated with hardware in future to automate the parking entry exit barrier.
- Payment mode can be made more secured.

7.4 ACHIEVEMENT OF THE PROJECT

1. Clients can create accounts on the system through registration .
2. System Administrator can manage the Employees by creating for them the account
3. Employees are able to book the parking for the client
4. Clients can view the parking available and reserve parking lot online.
5. System Administrators can manage the parking lot, transaction and also, he can manage the client.
6. System Administrators can efficiently and effectively manage all the users on the application and roles.

7.5 REFERENCE

1. P. White, “No Vacancy: Park Slopes Parking Problem and How to Fix It,” http://www.transalt.org/

2. “Solutions for improving city operatio,” http://www.streetlinenetworks.com/site/index.php

3. Y. Peng, Z. Abichar, and J. Chang, “Roadside-aided routing (RAR) in vehicular networks”, in Proc. IEEE ICC ’06, Vol. 8, pp. 3602-3607, Istanbul, Turkey, June 2006.

4. “Open Spot,” http://openspot.googlelabs.com/

5. R. Charette, “Smart Parking Systems Make It Easier to Find a Parking Space,” http://spectrum.ieee.org/green -tech/advancedcars/ smart-parking-systems-make-it- easier-to-find-a parking- space/0, 2007.

6. R. Lu, X. Lin, H. Zhu and X. Shen, “SPARK: A New VANET-based Smart Parking Scheme for Large Parking Lots,” in Proceedings of IEEE NFOCOM’07, 2007.

7. W. Mao, Modern Cryptography: Theory and Practice, Prentice Hall PTR, 2003.

8. D. Cook, S. Das, Smart Environments: Technologies, Protocols, and Applications, John Wiley, 2004.
9. M. Caliskan, D. Graupner and M. Mauve, “Decentralized Discovery of Free Parking Places,” in Proc. of the Third ACM International Workshop on Vehicular Ad Hoc Networks (VANET 2006),, 2006.

10. H. Varian, Microeconomic Analysis, New York: Norton, 2003