# ABSTRACT

Nowadays the traffic problem was rabidly increasing urban areas. So we can findthe solution for this problem. In this paper we proposed the one solution for reducing thisProblem. So, in this paper focus on smart android based parking control application, and which will helps find the nearest parking area of our current location. In this application will Helps the pay process of parking. After the car is being parked, a timer will start to count theAmount of time the car is spent in the parking lot. When the car owner wants to release the parking slot, he/she will take away the car and the timer will be stopped. The Smart Parking System project aims to revolutionize traditional parking methods by integrating full stack development and Android technology. This system enhances parking efficiency by providing real-time information on parking space availability, allowing users to locate and reserve parking spots conveniently through a user-friendly Android application. The full stack development ensures seamless communication between the front-end Android interface and the back-end server, enabling data synchronization and efficient management of parking resources. By leveraging technology to optimize parking operations, this project contributes to reducing traffic congestion, saving time for users, and improving overall urban mobility.

## CONTENTS

# LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

1. **Overview of the Project:** The Smart Parking System project aims to revolutionize the parking experience by utilizing full stack development and Android technology to create an efficient and user-friendly parking solution.

## 2. Full Stack Development:

**Frontend Development:** Discuss the creation of an intuitive user interface for drivers to easily locate and reserve parking spots through the Android application.
**Backend Development:** Explain the backend system that manages parking spot availability, reservations, and payment processing to ensure a seamless parking experience.

## 3. Android Application:

Features: Detail the features of the Android app, such as real-time parking spot availability, navigation to the parking location, online reservations, and secure payment options.
**User Interface:** Describe the design elements that enhance user experience, including interactive maps, personalized profiles, and notifications for parking updates.

By incorporating full stack development techniques and Android technology, the Smart Parking System project aims to optimize parking management, improve user convenience, and promote efficient use of parking spaces.

## 1.1 Objective

The Smart Parking System project, developed using full stack technologies and Android, aims to revolutionize parking management in Kompalle, Telangana, India. The project's main goal is to optimize parking operations, enhance user experience, and improve resource utilization. By providing real-time parking availability, seamless reservation processes, and efficient payment options through the Android application, the project aims to streamline parking processes and reduce congestion in urban areas.

Additionally, the focus on scalability and sustainability ensures that the Smart Parking System can adapt to future needs and continue to offer efficient parking solutions for the community

## 1.2 Overview

The Smart Parking System project, developed using full stack technologies and Android, is designed to transform parking management in Kompalle, Telangana, India. This innovative system aims to optimize parking operations, enhance user experience, and improve resource utilization. By offering real-time parking availability, seamless reservation processes, and efficient payment options through the Android application, the project seeks to streamline parking procedures and alleviate congestion in urban areas. Emphasizing scalability and sustainability, the Smart Parking System is crafted to evolve with future demands, ensuring continued delivery of effective parking solutions for the community. The Smart Parking System project, developed using full stack technologies and Android, is designed to transform parking management in Kompalle, Telangana, India. This innovative system aims to optimize parking operations, enhance user experience, and improve resource utilization. By offering real-time parking availability, seamless reservation processes, and efficient payment options through the Android application, the project seeks to streamline parking procedures and alleviate congestion in urban areas. Emphasizing scalability and sustainability, the Smart Parking System is crafted to evolve with future demands, ensuring continued delivery of effective parking solutions for the community

# CHAPTER 2

# LITERATURE SURVEY

1. **Smart Parking Systems Research:**

Explore academic papers and research studies on smart parking systems to understand the evolution, benefits, and challenges associated with implementing technology-driven parking solutions.

2. **IoT in Parking Management:**

Investigate how Internet of Things (IoT) technology is utilized in parking management systems to enhance efficiency, optimize resource allocation, and improve user experience in parking facilities.

3. **Mobile Applications for Parking Solutions:**

Review literature on mobile applications designed for parking solutions to analyze user interfaces, functionalities, and the impact of mobile technology on parking convenience and accessibility.

4. **Full Stack Development in Parking Systems:**

Examine the role of full stack development in creating comprehensive parking systems, including backend server infrastructure, databases, APIs, and frontend interfaces to provide a seamless user experience.

5. **User Experience and Urban Environments:**

Look into studies focusing on user experience in parking applications and the influence of smart parking systems on urban environments, including traffic flow, parking space utilization, and sustainability aspects.

By delving into these specific areas within the literature, you can gather valuable insights and knowledge to inform the development and implementation of your Smart Parking System project using full stack and Android technologie

## 6. Case Studies and Best Practices:

Explore case studies highlighting successful implementations of smart parking systems using full stack and Android technologies to identify best practices, challenges faced, and lessons learned in real-world scenarios.

## 7. Integration of Machine Learning in Parking Systems:

Investigate how machine learning algorithms are integrated into parking systems to predict parking availability, optimize parking space allocation, and improve overall system efficiency.

## 8. Security and Privacy Considerations:

Review literature on security measures and privacy considerations in smart parking systems, including data protection, authentication mechanisms, and secure payment processing to ensure user information is safeguarded.

## 9. Sustainability and Environmental Impact:

Examine studies focusing on the environmental impact of smart parking systems, including reduced traffic congestion, lower emissions, and the promotion of sustainable transportation practices in urban areas.

## 10. Future Trends in Smart Parking Technology:

Look into emerging trends and future developments in smart parking technology, such as the integration of autonomous vehicles, smart city initiatives, and the potential impact of 5G technology on parking systems.

# CHAPTER 3

# SYSTEM ANALYSIS AND DESIGN

## 3.1 Existing System

For your Smart Parking System project utilizing full stack and Android technologies, you can explore existing systems and platforms that incorporate similar functionalities. Look into commercial smart parking solutions, open-source projects, and research prototypes that involve full stack development and Android applications for parking management. Analyze their features, architecture, user interfaces, and integration of technologies to gather insights and inspiration for designing and implementing your own smart parking system**.**

### 1. Current Challenges

**Limited Availability Information**: Most systems do not provide real-time availability of parking spaces, leading to frustration for users searching for parking.

**User Interface Issues:** Existing applications may have complex interfaces that are not user-friendly, making it difficult for users to navigate.

**Inefficient Payment Methods:** Many systems rely on outdated payment methods, which can lead to delays and inconvenience for users.

**Lack of Integration:** Current systems often do not integrate well with other services, such as navigation apps, making it harder for users to find parking spots.

### 2. Technology Stack

**Frontend:** Existing systems typically use basic web technologies (HTML, CSS, JavaScript) for their user interfaces, which may lack responsiveness and modern design principles.

**Backend:** Many systems are built on traditional server architectures without leveraging cloud technologies, limiting scalability and performance.

**Database Management:** Existing solutions often use relational databases that may not efficiently handle large datasets or real-time data processing.

### 3. Features of Existing Systems

**Basic Parking Reservation**: Some systems allow users to reserve parking spots but may not provide real-time updates on availability.

**Payment Processing:** Most systems offer payment processing, but the methods can be limited (e.g., cash only or outdated credit card system

**User Registration:** Existing systems typically require user registration but may not offer social media login options for convenience.

**4. Conclusion**

While existing smart parking systems have made strides in improving parking management, they still face significant challenges that can be addressed through a full-stack development approach combined with an Android application. By focusing on real-time data, user-friendly interfaces, efficient payment methods, and better integration with other services, a new smart parking system can greatly enhance the overall user experience.

## 3.2 PROPOSED SYSTEM

developing a Smart Parking System using full stack and Android technologies involves creating a user-friendly app for parking spot reservations, managing parking data, integrating payment systems, and ensuring real-time updates for users. The system should include features like mapping, notifications, and an admin panel for operators to monitor bookings and system performance. By incorporating these functionalities, your project can offer a seamless and efficient parking solution for users. Let me know if you need more details or guidance on any specific aspect of the project!

**1. Key Features of the Proposed System**

**Real-Time Availability Tracking**

- The system will provide real-time information on parking space availability using IoT sensors and GPS technology. Users can see which spots are free before they arrive.

**User-Friendly Interface**

- The Android application will feature an intuitive and modern user interface, making it easy for users to navigate, search for parking spots, and complete transactions.

**Reservation System**

- Users can reserve parking spots in advance through the mobile app. This feature will allow users to guarantee their space, reducing time spent searching for parking.

**Efficient Payment Processing**

- The system will support multiple payment options, including mobile wallets, credit/debit cards, and QR code payments, ensuring convenience and speed during the payment process.

**Integration with Navigation Apps**

- The application will integrate with popular navigation services (like Google Maps) to provide users with directions to their reserved parking spots, enhancing the overall user experience.

**User Authentication**

- The system will allow users to register using their email, phone number, or social media accounts, providing flexibility and ease of access.

**2. Technology Stack**

**Frontend**

**Android:** The mobile application will be developed using Kotlin or Java for Android, ensuring compatibility and performance on various devices.

**Backend**

**Node.js**: The backend will be built using Node.js for its scalability and efficiency in handling multiple requests.
**Express.js:** This framework will be used to build RESTful APIs for communication between the frontend and backend.

**Database**

**MongoDB:** A NoSQL database will be used to store user data, parking space information, and transaction records, allowing for flexibility and scalability.

**IoT Integration**

**Sensors:** Parking sensors will be deployed to detect the occupancy of parking spaces and send real-time data to the server.

# CHAPTER 4

# SOFTWARE REQUIREMENTS AND SPECIFICATION

## 4.1 Design

### 4.1.1 System Architecture

designing the system architecture for your Smart Parking System project using full stack and Android involves setting up a frontend Android application for user interactions, a backend server for business logic and data handling, APIs for communication, cloud services for scalability, security measures, real-time updates, payment gateway integration, and monitoring tools. By structuring these components effectively, you can ensure a secure, efficient, and user-friendly Smart Parking System. Let me know if you need more details or guidance on any specific part of the architecture!

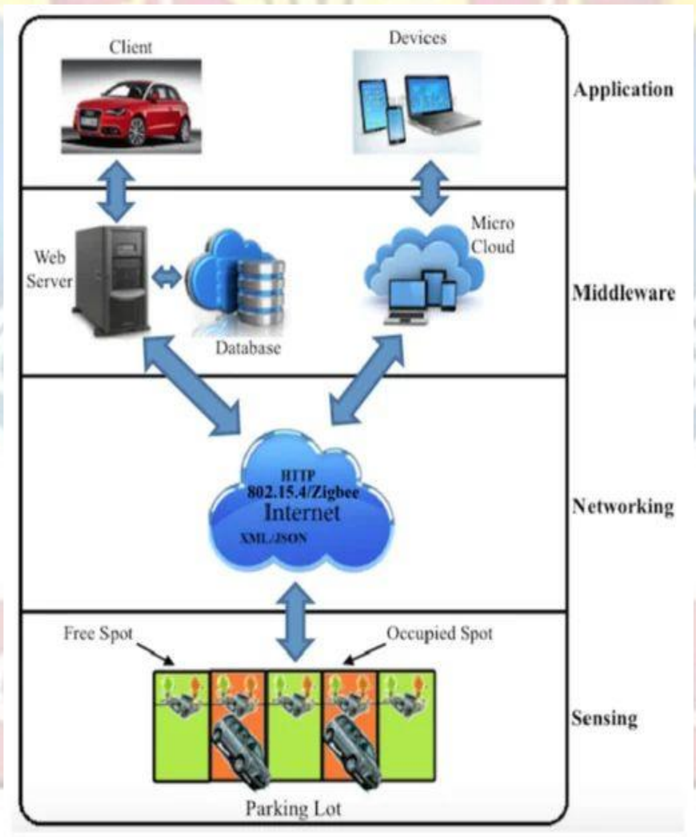


**Figure 4.1.1 SystemArchitecture**

**Description**:figure 4.1.1 visually presents the overall structure and organization of the server, database, web interface, and mobile app components, depicting how they interact and function together to provide the smart parking solution seamlessly.

## Class Diagram

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

The upper part holds the name of the class

The middle part contains the attributes of the class

The bottom part gives the methods or operations the class can take or undertake

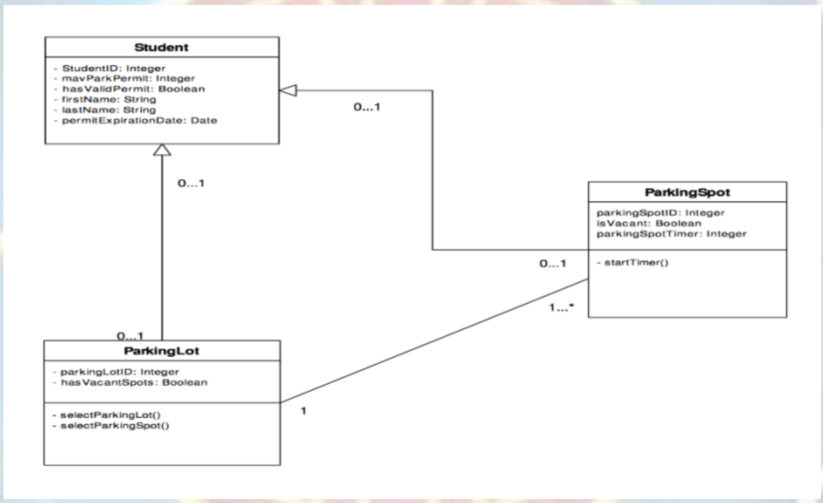


**Fig 4.1.2 class Diagram**

**Description:** figure 4.1.2 illustrates the structure of the system by detailing the classes, their attributes, methods, and relationships, providing a blueprint for the implementation of the server, database, web interface, and mobile app components.

## 4.1.2 Use case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

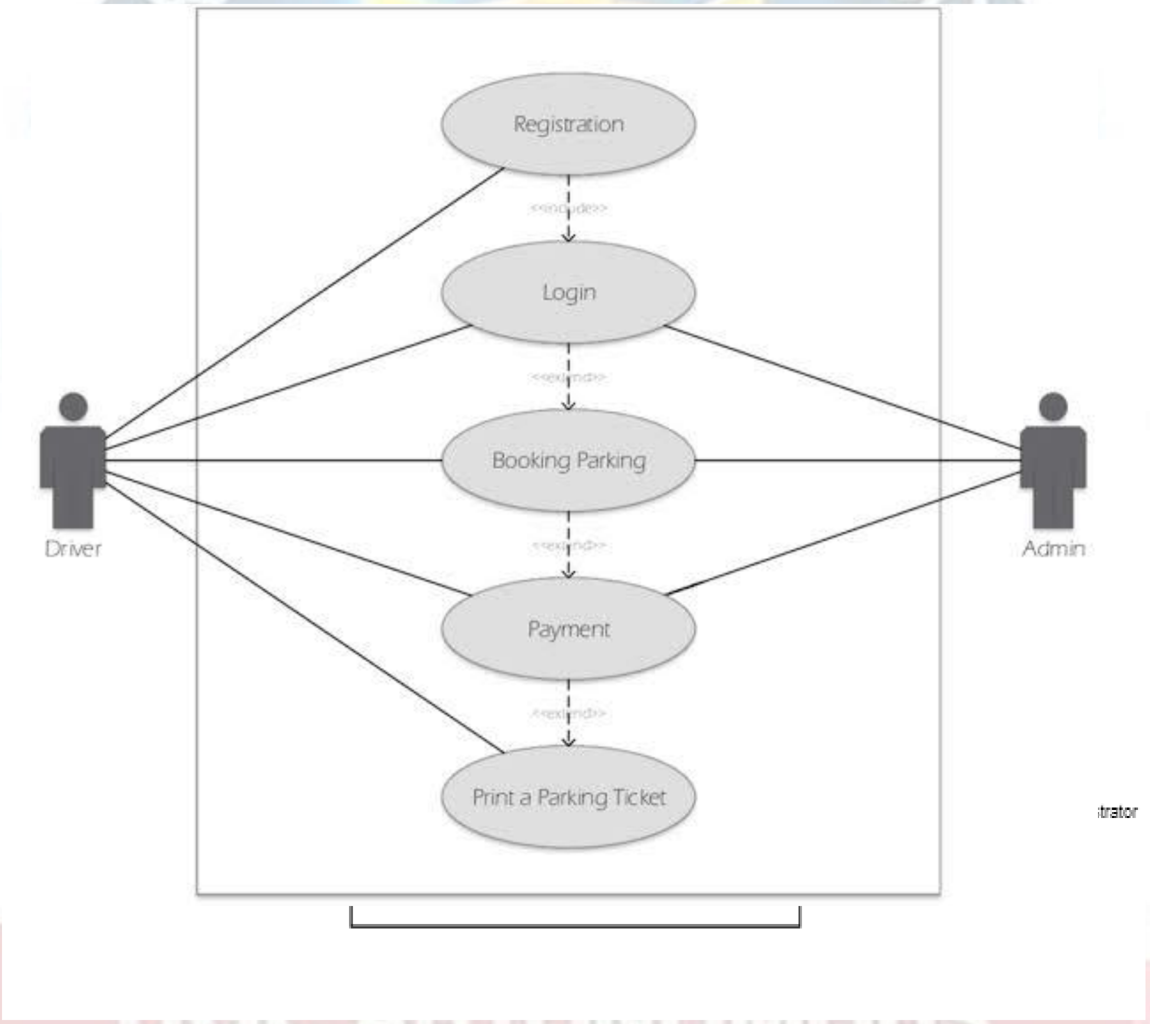


**Fig 4.1.3 Use case Diagram**

**Description:** figure 4.1.3 showcases the various functionalities and interactions between administrators, And the system components like the server, database, web interface, and mobile app, defining the different roles and actions within the system.

## 4.1.3 Sequence Diagram

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

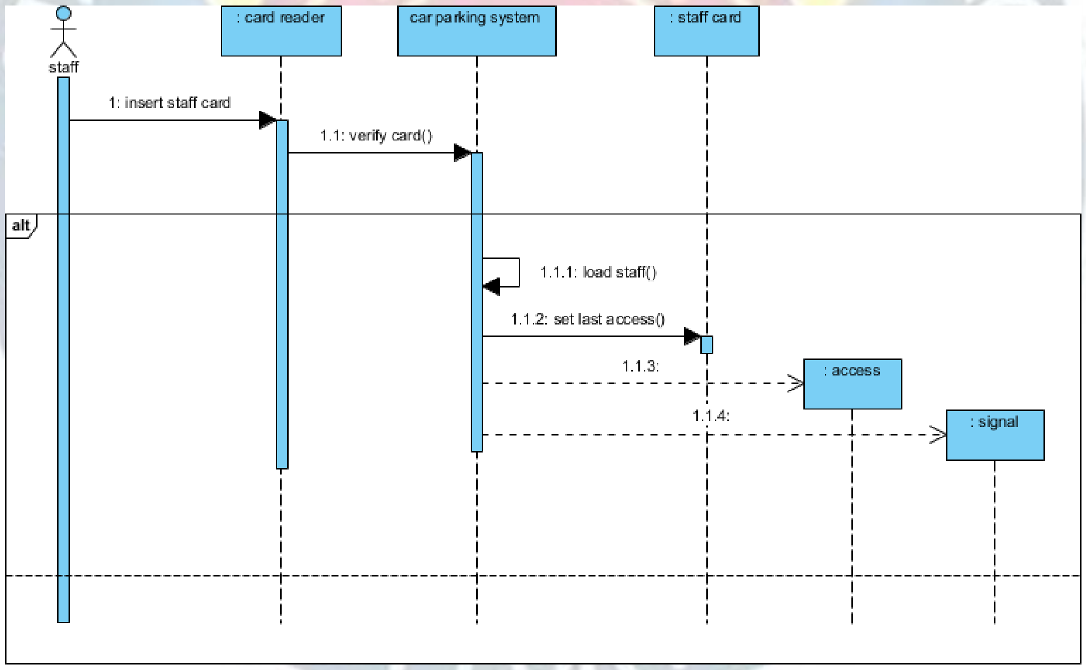


**Fig 4.1.4 Sequence Diagram**

**Description**:figure 4.1.4 represents outlines the step-by-step interactions between the server, database, web interface, and mobile app components, illustrating the order of messages exchanged during the system's operation.

## 4.1.4 Collaboration Diagram

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system.

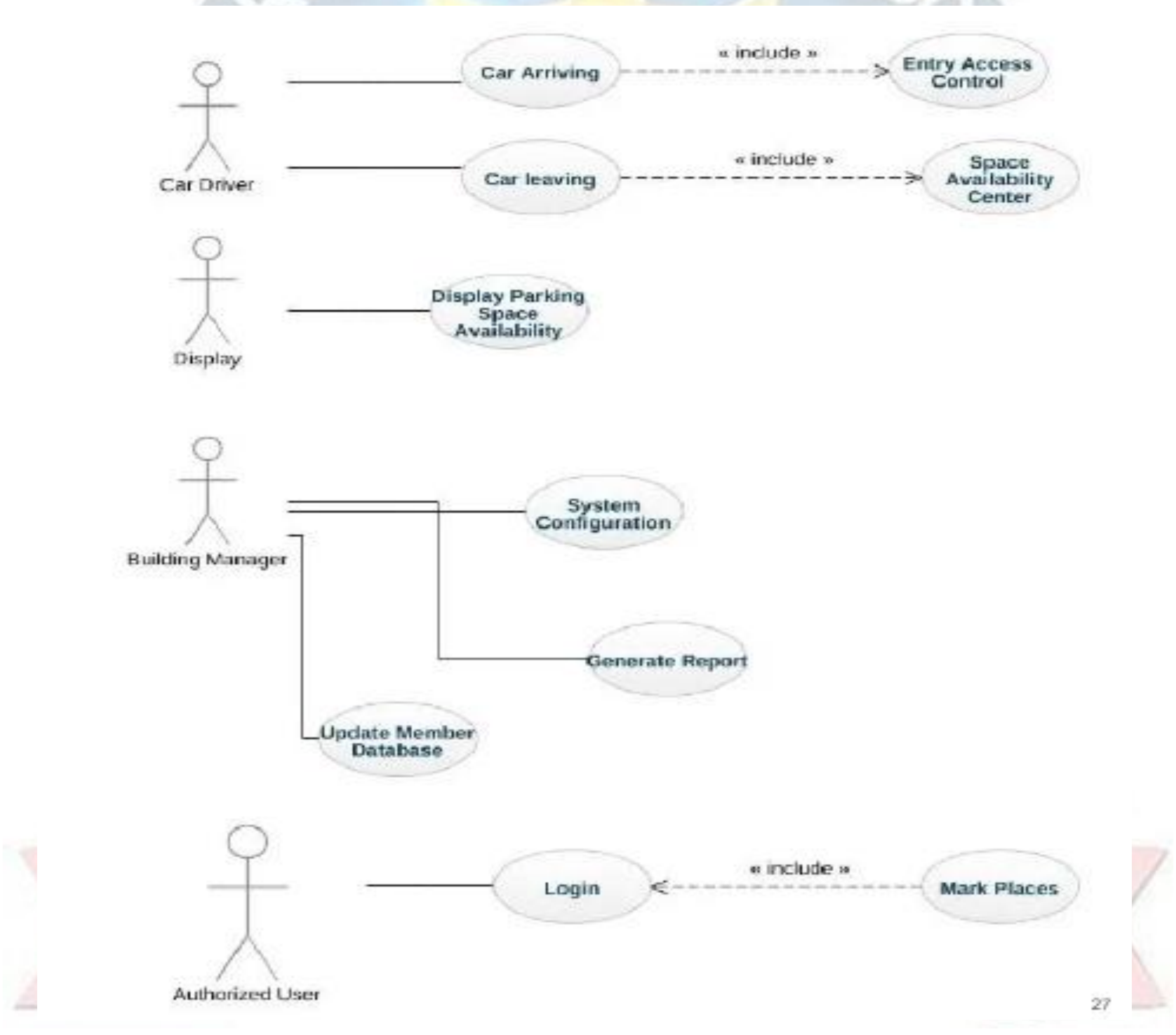


**Fig 4.1.5 Collaboration Diagram**

**Description:** figure 4.1.5 visually represents how the server, database, web interface, and mobile app collaborate and communicate in real-time, showcasing the interactions between the different components in the system.
.

### 4.1.5 Deployment Diagram

A deployment diagram in the Unified Modeling Language models the physical deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

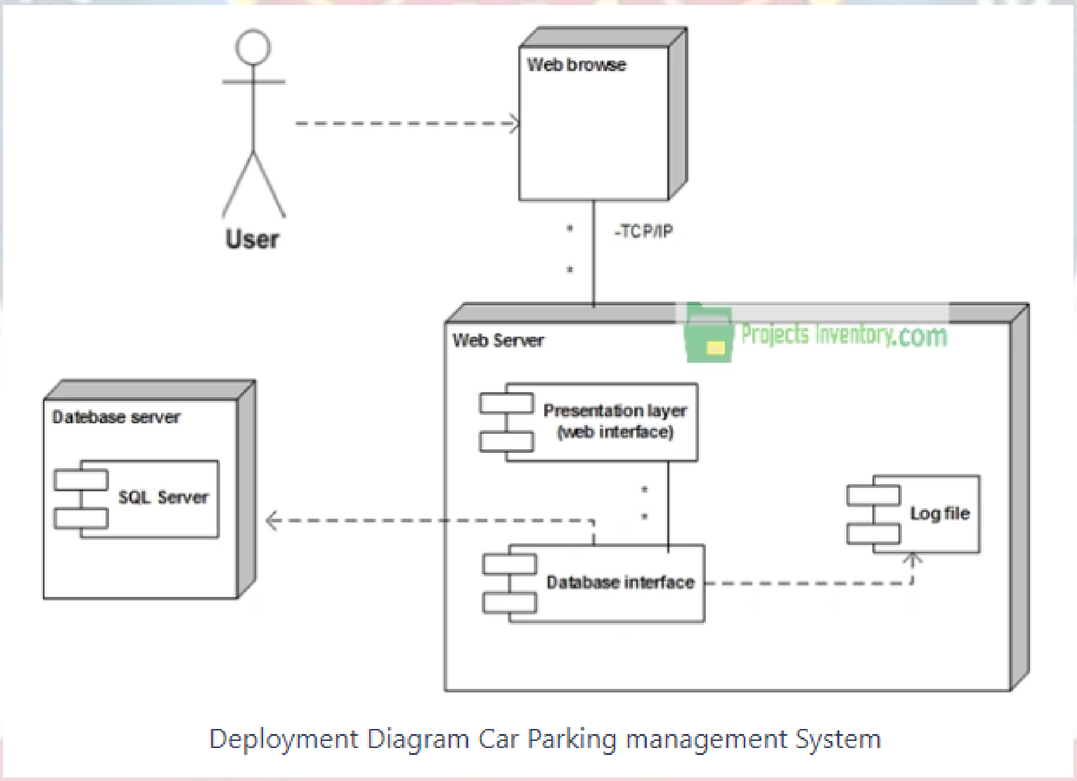


**Fig 4.1.6 Deployment Diagram**

**Description:** figure 4.1.6 illustrates how the server, database, web interface, and mobile app components are distributed and interact within the system, showing the deployment configuration and communication pathways.

## 4.1.6 Data Flow Diagram

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.
As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

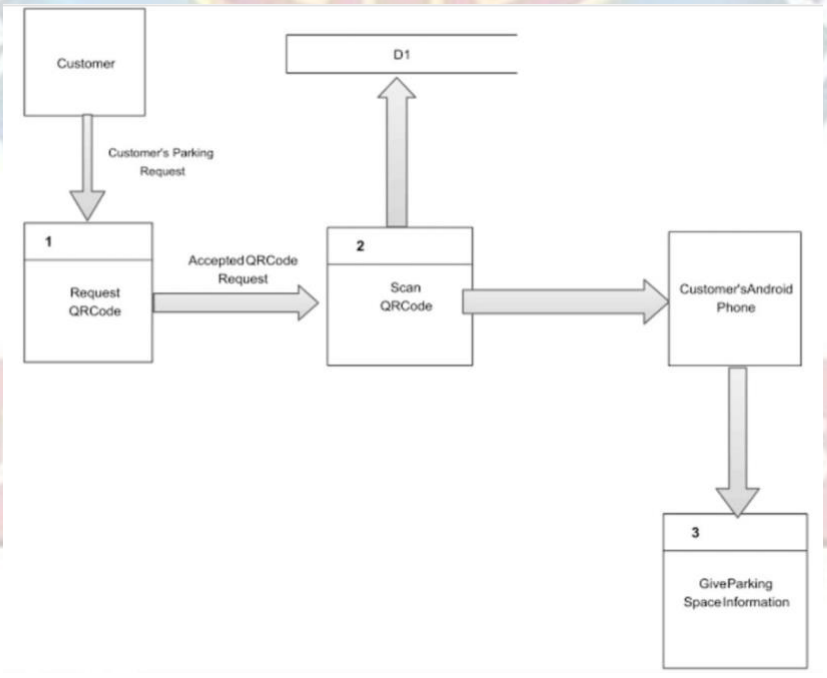


**Fig 4.1.7 Data Flow Diagram**

**Description**:figure 4.1.7 represents the tasks done by the components involved in the project, how information moves between the server, database, web interface, and mobile app in real-time.

## 4.1 Modules

1. Android App
2. Backend Server
3. API’s

### 4.1.2 Modules Description

### 1. Android App:

**Finding Parking Spots:**

- Display available parking spots on the map.

- Allow users to search for parking spots based on location or proximity.

**Reservation System:**

- Enable users to reserve parking spots in advance.

- Implement a booking system with date and time selection.

**Navigation Features:**

- Integrate GPS navigation to guide users to their reserved parking spot.

- Provide directions within the app for seamless navigation.

### 2. Backend Server:

**Data Processing:**

- Manage user reservations, parking spot availability, and transaction details.

**Communication Handling:**

- Facilitate seamless communication between the Android app and the database.

**Database Management:**

- Store and retrieve parking spot information, user details, and reservation data.

### 3. API’S:

**Functionality Integration:**

- Enable the Android app to interact with the backend server seamlessly.

**Data Exchange:**

- Facilitate the transfer of data between the frontend app and the backend server.

**Scalability and Flexibility:**

- Design APIs that allow for future expansion and integration of new features.

## 4.2 System Requirements

### 4.2.2 HARDWARE REQUIREMENTS:

Hard disk: 250GB or above

Processor: i3 or above

Ram: 4GB or above

### 4.2.3 SOFTWARE REQUIREMENTS:

Operating system: windows 7 or above (64bit)

Java: 1.8

Net beans: 8.0.2

MySQL: 5.0

Eclipse

### 4.2.4 History of Java:

Java was developed by James Gosling and his team at Sun Microsystems in the early 1990s.Originally named Oak, it was designed for consumer electronics but later shifted its focus to internet programming.

### 4.2.5 Why Java was created?

Java is a widely-used programming language for coding web applications. It has been a popular choice among developers for over two decades, with millions of Java applications in use today. Java is a multi-platform, object-oriented, and network-centric language that can be used as a platform in itself.

### 4.2.6 Features of Java:

- Simple and Easy to Learn.
- Object-Oriented Programming.
- Platform Independence.
- Automatic Memory Management.
- Security.
- Rich API.
- Multithreading.
- High Performance.

## 4.3 Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

**Types of testing**

### 4.3.2 Unit Testing

**Purpose of Unit Testing:**

- Unit testing is a software testing technique where individual units or components of a program are tested in isolation.

- The primary goal is to validate that each unit of code performs as expected and functions correctly.

**Implementation Process:**

- Developers write test cases for specific units of code, often focusing on small functions or methods.

- These tests are automated and run frequently during development to catch bugs early.

**Benefits of Unit Testing:**

- Helps identify defects early in the development cycle, reducing the cost of fixing issues.

- Improves code quality, readability, and maintainability by encouraging modular and well-structured code.

**Tools and Frameworks:**

- Various tools and frameworks like JUnit, TestNG, and Mockito are commonly used for Java unit testing.

Unit testing involves testing individual components or units of a software application to ensure they function as expected. Each unit is tested in isolation from the rest of the application.

**Application in Blockchain Project:**

**Frontend Development (Android):**

- Develop an Android application for user interaction.
- Use Android Studio with Java or Kotlin for frontend design.

**Backend Development (Full Stack):**

- Create a backend server to manage data and handle requests.
- Use Node.js or Java Spring Boot for backend logic.

### 4.3.3 Functional Testing

Functional testing involves verifying that the software performs its intended functions. It focuses on the functionality of the system as specified in the requirements.

**Test Planning:**

- Define the scope of testing, including features like parking spot reservation, real-time updates, and user notifications.

- Create test cases covering both frontend (Android) and backend functionalities.

**Test Execution:**

- Execute test cases to validate the functionality of the Android application and backend server.

- Verify that the Android app interacts correctly with the backend system.

- Test features like user registration, booking a parking spot, and receiving real-time updates.

### 4.3.4 White Box Testing

White box testing involves testing the internal structures or workings of an application. It requires knowledge of the code, architecture, and internal processes of the system.

**Code Review:**

- Conduct a thorough review of the source code for both the backend (full-stack) and Android components.
- Check for coding standards, best practices, and potential vulnerabilities.

**Unit Testing:**

- Write and execute unit tests for individual functions and modules in the backend server and Android app.
- Test the logic, data processing, and interactions within each component.

**Integration Testing:**

- Perform integration tests to ensure that the backend and frontend components work together seamlessly.
- Validate data flow, API communication, and system integration.

**Code Coverage Analysis:**

- Use code coverage tools to measure the percentage of code that is executed during testing.
- Aim for high code coverage to ensure comprehensive testing of the system.

### 4.3.5 Black Box Testing

Black box testing involves testing the functionality of the application without knowledge of the internal workings of the system. It focuses on input and output rather than the code itself.

**Functional Testing:**

- Test the functionality of the Smart Parking System without looking at the internal code.
- Verify that users can perform actions like finding parking spots, reserving spots, and making payments successfully.

**User Interface Testing:**

- Evaluate the user interface of the Android app and web interface (full-stack) for usability and consistency.
- Ensure that the interfaces are intuitive, responsive, and user-friendly.

**Compatibility Testing:**

- Test the Smart Parking System on different devices, browsers, and operating systems to ensure compatibility.
- Check for any issues related to device-specific functionalities.

### 4.3.6 System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to use the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

### 4.3.7 Module Testing

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

### 4.3.8 Integration Testing

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

### 4.3.9 Acceptance Testing

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

# CHAPTER 5

# SOURCE CODE

Java:

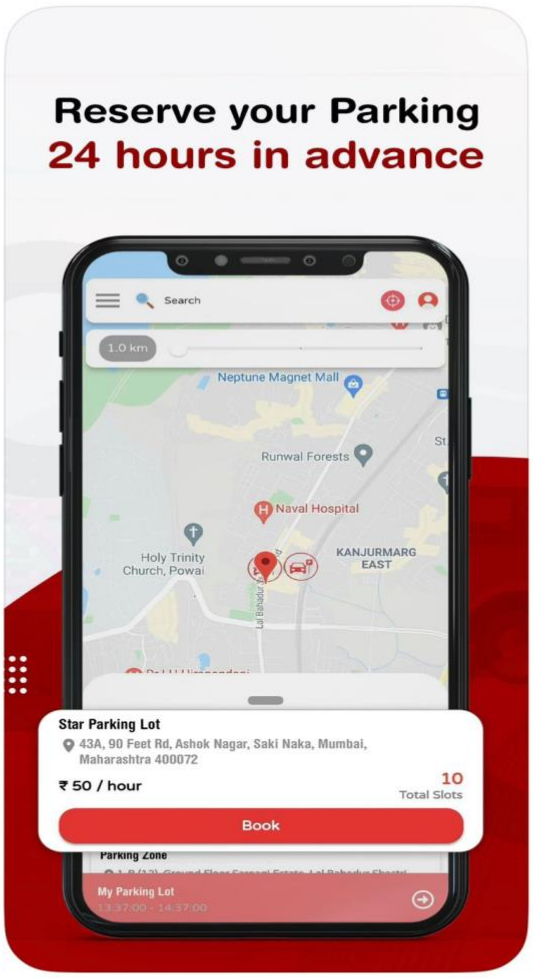
```
/*
 * To change this license header,
choose License Headers in Project
Properties.
 * To change this template file,
choose Tools | Templates
 * and open the template in the
editor.
 */
package New;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;

/**
 *
 * @author Lenovo
 */
@WebService(serviceName =
"Parkingservice")
public class Parkingservice {
```

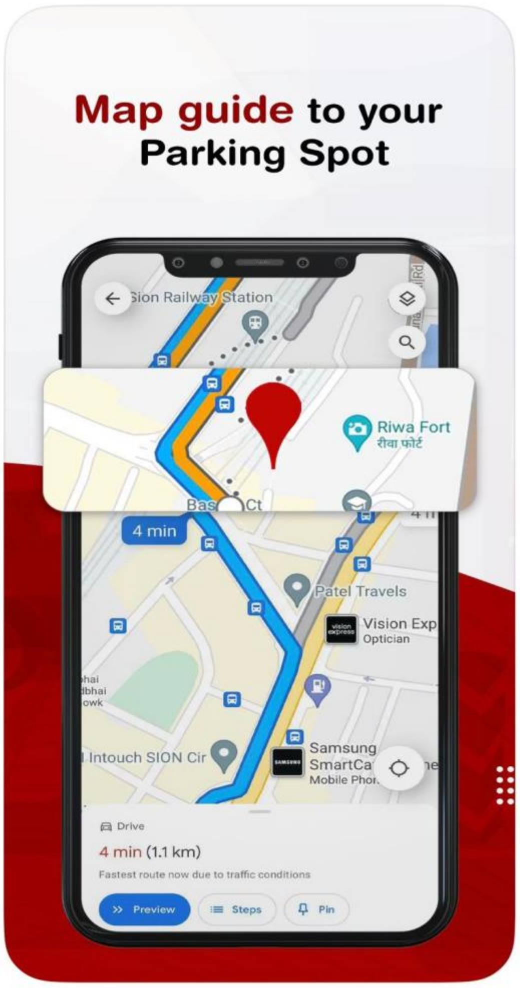
```
* @return
     */
   @WebMethod(operationName = "Register")
  public String Register(@WebParam(name = "username") String username, @WebParam(name =
"password") String password, @WebParam(name = "email") String mail, @WebParam(name = "mobile")
String mobile) {
    try
    {
    Class.forName("com.mysql.jdbc.Driver");
    Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/newpark","root","password");
    PreparedStatement ps=con.prepareStatement("Insert into user(username,password,mail,mobile)
Values(?,?,?,?)");
    ps.setString(1, username);
    ps.setString(2, password);
    ps.setString(3, mail);
    ps.setString(4, mobile);
    ps.executeUpdate();
    }
        catch(ClassNotFoundException | SQLException e)
        {
        System.out.println(e);
        }
     return "success";


    }
@WebMethod(operationName = "Login")
  public String Login(@WebParam(name = "user_name") String user_name, @WebParam(name =
"password") String password) {
   try{
Class.forName("com.mysql.jdbc.Driver");
     Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/newpark","root","password");
     Statement st=con.createStatement();
    ResultSet rs=st.executeQuery("Select * from user where username='"+user_name+"' and
password='"+password+"' ");
    if(rs.next())
    {
return "success";


    }
    else{
   return "username and password invalid";
    }


   }
      catch(Exception e){
      e.printStackTrace();
      }
      return "internal server error";


 }

  @WebMethod(operationName = "getDeviceList1")
    public String getDeviceList1 (@WebParam(name = "from") String vegitable,String method) {


    StringBuilder sb = new StringBuilder();
  try{
          Class.forName("com.mysql.jdbc.Driver");
          Connection
conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/newpark","root","password");
          Statement st=conn.createStatement();
          ResultSet rs=st.executeQuery("select * from slots ");
          while(rs.next())
          {
             String place=rs.getString("place");
             String slots=rs.getString("slots");
             String price=rs.getString("price");
sb.append("Place: ").append(place);
             sb.append("\n");
             sb.append("AvailableSlots: ").append(slots);
             sb.append("\n");
             sb.append("Price: ").append(price);
             sb.append("@");

           }

   }
     catch(ClassNotFoundException | SQLException ex){
       //Logger.getLogger(Registration.class.getName()).log(Level.SEVERE,null,ex);
       return "server temporarily not avilable";
     }
      System.out.println(sb.toString());
      return sb.toString();

    }
```

```
      @WebMethod(operationName = "BankDetails")
  public String BankDetails(@WebParam(name = "name") String name, @WebParam(name =
"accountno") String accountno, @WebParam(name = "cvv") String cvv, @WebParam(name =
"expirydate") String expirydate,@WebParam(name = "password") String password) {
    try{



     Class.forName("com.mysql.jdbc.Driver");
      Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/newpark","root","password");
     PreparedStatement ps=con.prepareStatement("Insert into
bankdetails(name,accountno,cvv,expirydate,password)
Values('"+name+"','"+accountno+"','"+cvv+"','"+expirydate+"','"+password+"')");
    ps.executeUpdate();
        }catch(Exception e){
        e.printStackTrace();
        }



        return "success";

    }


  @WebMethod(operationName = "BLogin")
  public String BLogin(@WebParam(name = "user_name") String user_name, @WebParam(name =
"password") String password) {
    try{
     Class.forName("com.mysql.jdbc.Driver");
      Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/newpark","root","password");
      Statement st=con.createStatement();
     ResultSet rs=st.executeQuery("Select * from bankdetails where name='"+user_name+"' and
password='"+password+"' ");
     if(rs.next())
     {
         return "success";


     }
     else{
    return "username and password invalid";
     }



    }
        catch(Exception e){
        e.printStackTrace();
            System.out.println(e);
        }
        return "internal server error";

  }



  @WebMethod(operationName = "Sms")
  public String Sms(@WebParam(name = "phone") String phno) {
    try{
     Class.forName("com.mysql.jdbc.Driver");
      Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/newpark","root","password");
     Statement st=con.createStatement();
     ResultSet rs=st.executeQuery("Select * from user where mobile='"+phno+"'");
     if(rs.next())
     {
         return "success";


     }
     else
     {
    return "mobile number is invalid";
     }
    }
      catch(Exception e){
        e.printStackTrace();
              System.out.println(e);
        }
        return "internal server error";
    }//
  }
```

# CHAPTER 6
# EXPERIMENTAL RESULTS

## APPLICATION INTERFACE:





**Description:**

**IMAGE 1:** As you can see user can book his parking slot in advance

**IMAGE 2:**The ANDROID APPLICATION navigates user to the parking slot

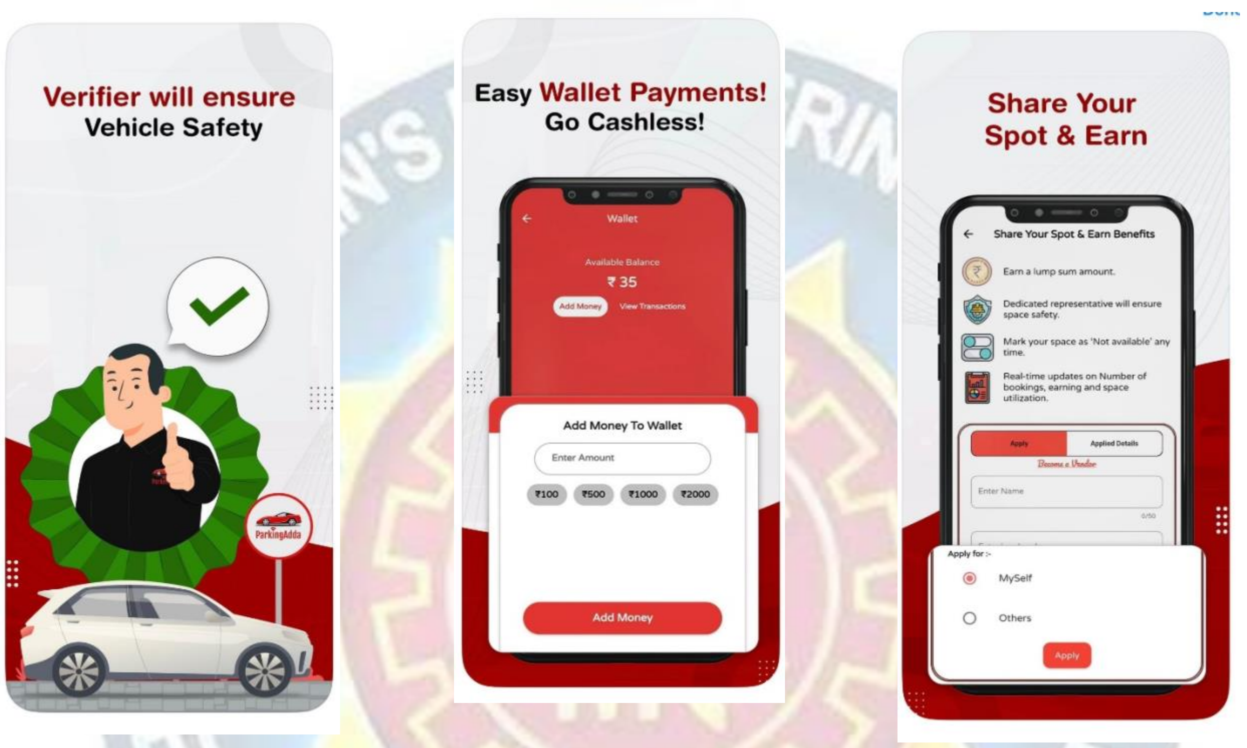

Verifier will ensure Vehicle Safety
ParkingAdda
Easy Wallet Payments! Go Cashless!
Wallet
Available Balance
₹ 35
Add Money
View Transactions
Add Money To Wallet
Enter Amount
₹100
₹500
₹1000
₹2000
Add Money
Share Your Spot & Earn
Share Your Spot & Earn Benefits
Earn a lump sum amount.
Dedicated representative will ensure space safety.
Mark your space as 'Not available' any time.
Real-time updates on Number of bookings, earning and space utilization.
Apply
Applied Details
Become a Vendor
Enter Name
0/50
Apply for :-
MySelf
Others
Apply

**Fig 6.1 HomePage**

**Description:**

**IMAGE 3:**Verifier(PARKING SLOT OWNER)will ensure Vehcile's Safety

**IMAGE 4:**Self wallet is provided for user convenience

**IMAGE 5:** Anyone who is having parking place can share their spot and EARN

## FRONT END USER LOGIN/SIGNUP PAGE

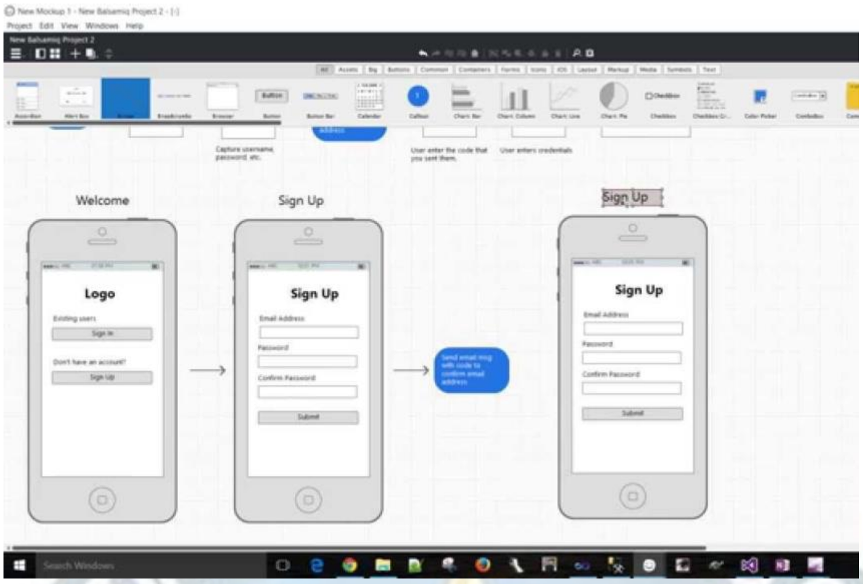
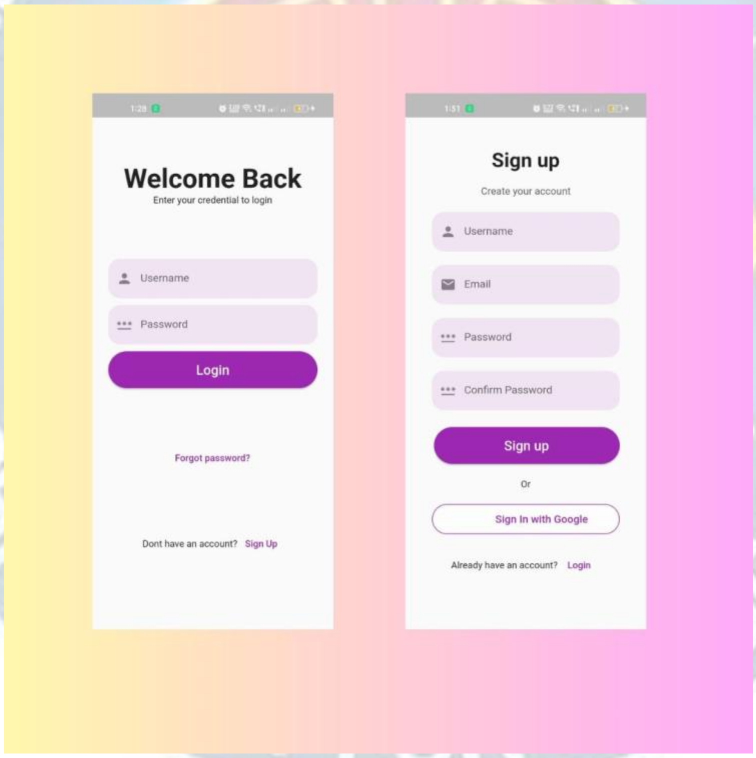


**Fig 6.2 USER Signup/Login**

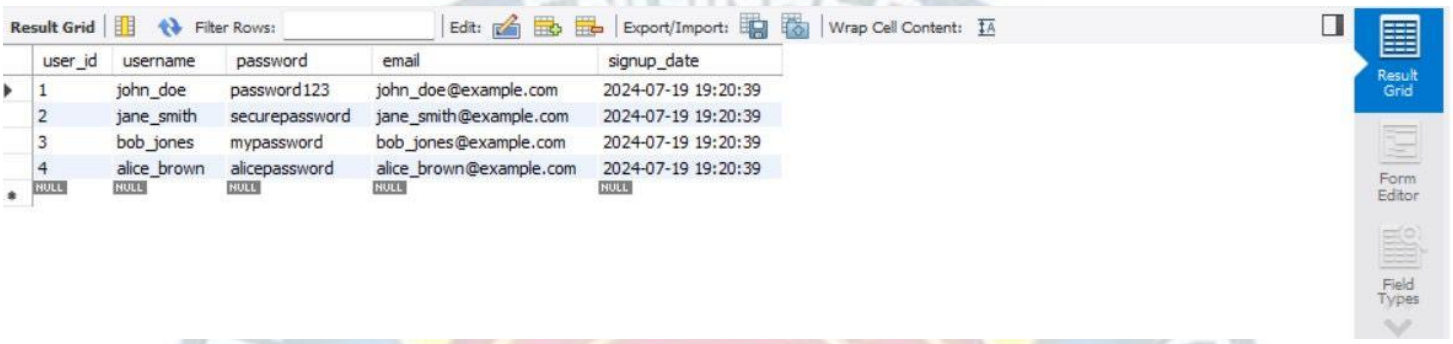


**Description**:figure 6.2 represents:

WELCOMEBACK: is for already existing users

SIGNUP: is for new users

## BACKEND SQL DATABASE



| user_id | username | password | email | signup_date |
|---|---|---|---|---|
| 1 | john_doe | password123 | john_doe@example.com | 2024-07-19 19:20:39 |
| 2 | jane_smith | securepassword | jane_smith@example.com | 2024-07-19 19:20:39 |
| 3 | bob_jones | mypassword | bob_jones@example.com | 2024-07-19 19:20:39 |
| 4 | alice_brown | alicepassword | alice_brown@example.com | 2024-07-19 19:20:39 |
| NULL | NULL | NULL | NULL | NULL |

**Fig 6.3 SQL DataBase**

<u>USE OF SQLDATABASE IN THIS PROJECT</u>: In our smart parking system project that combines full stack development with Android, using an SQL database is crucial for storing and managing data efficiently. SQL databases allow you to organize and retrieve information related to parking spots, availability, reservations, user details, and transactions seamlessly. By utilizing SQL, we can create structured queries to interact with the database, ensuring smooth communication between the backend systems and the Android application.

**Description:** figure 6.3 represents User's login and signup information is stored in our SQL DATABASE.whenever a user enters into the application the credentials of the user is stored in the database

# CHAPTER 7
# CONCLUSION AND FUTURE ENHANCEMENT

## 7.1 Conclusion

Thus we proposed and implemented an app based parking reservation system which facilitates the user to book a parking slot. It also searches for the nearest parking slot and includes features like QR code scanning in order to ensure consistency. At last the online payment option eliminates the need for human intervention and making the system automated in real sense. Also the proposed system is very economical and easy to implement as it does not involve any expensive hardware or devices.

## 7.2 Future Enhancement

The proposed system can further be upgraded by including features like sensors and CCTV camera so that the latest status of the parking can be checked by the user through the same application, rather than providing statistical data(in the current system). Also, navigation facility can be added to guide the user at each step to make sure that the user reaches the destination slot in the minimum possible time.

# REFERENCES

[1] V. G. Sahu, V. Gulhane, and N. Shelokar, \A web based centralized vehicle parking system using gsm security," IJAIEM, vol. 2, no. 4, 2013.

[2] Z. Ji, I. Ganchev, M. O'Droma, L. Zhao, and X. Zhang, \A cloud- based car parking middleware for iot-based smart cities: Design and implementation," Sensors, vol. 14, no. 12, pp. 22 372{22 393, 2014.

[3] V. P. Bilodeau, \Intelligent parking technology adoption," Ph.D. dissertation, University of Southern Queensland, 2010.

[4] Y. Zhao and E. G. Collins Jr, \Robust automatic parallel parking in tight spaces via fuzzy logic," Robotics and Autonomous Systems, vol. 51, no. 2-3, pp. 111{127, 2005. [5]B. R. T. Authority, \Number of registered mo-tor vehicles in bangladesh (yearwise)," 2017, [On-line; accessed 15-September-2017]. [Online]. Avail-able: p://www.brta.gov.bd/newsite/en/whole-bangladesh-up- to-sep-2016/

[5] "Android Programming: The Big Nerd Ranch Guide" by Bill Phillips and Brian Hardy.

[6] "Full Stack Web Development with React and Node" by Shama Hoque.

[7] "Smart Parking Guidance System Using IoT" by Shubham Jadhav and Prof. G. N. Patil.

[8] "Android Studio Development Essentials" by Neil Smyth.

[9] "Full Stack Development with JHipster" by Deepu K Sasidharan and Sendil Kumar N.

[10] "Designing Smart Parking Management System using IoT" by M. S. Bheemalingeswara and Dr. R. N. Suresh.

[11] "Android Programming: Pushing the Limits" by Erik Hellman.