VEHICLE PARKING

A PROJECT REPORT

Submitted by

DENI SEBASTA R

In partial fulfilment for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

K. RAMAKRISHNAN COLLEGE OF ENGINEERING (AUTONOMOUS)
SAMAYAPURAM, TRICHY

ANNA UNIVERSITY, CHENNAI 600025

JUNE 2025

VEHICLE PARKING

Submitted by

DENI SEBASTA R (8115U23EC015)

in partial fulfillment of requirements for the award of the course

EGB1221 – DATABASE MANAGEMENT SYSTEM

in

ELECTRONICS AND COMMUNICATION ENGINEERING

Under the Guidance of

Mrs.R.Nalini

Department of Computer Science and Engineering

K. RAMAKRISHNAN COLLEGE OF ENGINEERING

K. RAMAKRISHNAN COLLEGE OF ENGINEERING
(AUTONOMOUS)
Under
ANNA UNIVERSITY, CHENNAI

K. RAMAKRISHNAN COLLEGE OF ENGINEERING (AUTONOMOUS)

Under

ANNA UNIVERSITY, CHENNAI

BONAFIDE CERTIFICATE

Certified that this project report on **"VEHICLE PARKING"** is the bonafide work of **DENI SEBASTA R (8115U23EC015)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

SIGNATURE	**SIGNATURE**
Dr. T. M. NITHYA, M.E., Ph.D.,	Mrs.R.NALINI, M.E., (Ph.D).,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
ASSOCIATE PROFESSOR	ASSISTANT PROFESSOR
Department of CSE	Department of CSE
K. Ramakrishnan College of Engineering (Autonomous)	K. Ramakrishnan College of Engineering (Autonomous)
Samayapuram – 621 112.	Samayapuram – 621 112.

Submitted for the viva-voce examination held on …………….

INTERNAL EXAMINER EXTERNAL EXAMINER

K. RAMAKRISHNAN COLLEGE OF ENGINEERING

(AUTONOMOUS)

Under

ANNA UNIVERSITY, CHENNAI

DECLARATION BY THE CANDIDATE

I declare that to the best of my knowledge the work reported here in has been composed solely by me and that it has not been in whole or in part in any previous application for a degree.

Submitted for the project Viva- Voce held at K. Ramakrishnan College of Engineering on ______________________

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution **“K.Ramakrishnan College of Engineering (Autonomous)”**, for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN, B.E.,** for having provided for the facilities during the course of our study in college

I would like to express our sincere thanks to our beloved Executive Director **Dr.S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. D. SRINIVASAN, B.E, M.E., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. T. M. NITHYA, M.E.,Ph.D.,** Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I am grateful to **Mrs. R. NALINI, M.E., (Ph.D)** Assistant Professor, **COMPUTER SCIENCE AND ENGINEERING**, K. Ramakrishnan College of Engineering (Autonomous), for her guidance and valuable suggestions during the course of study.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To achieve a prominent position among the top technical institutions.

MISSION OF THE INSTITUTION

- M1: To bestow standard technical education par excellence through state of the art infrastructure, competent faculty and high ethical standards.
- M2: To nurture research and entrepreneurial skills among students in cutting edge technologies.
- M3: To provide education for developing high-quality professionals to transform the society.

VISION OF DEPARTMENT

To create eminent professionals of Computer Science and Engineering by imparting quality education.

MISSION OF DEPARTMENT

M1: To provide technical exposure in the field of Computer Science and Engineering through state of the art infrastructure and ethical standards.

M2: To engage the students in research and development activities in the field of Computer Science and Engineering.

M3: To empower the learners to involve in industrial and multi-disciplinary projects for addressing the societal needs.

PROGRAM EDUCATIONAL OBJECTIVES

Our graduates shall

PEO1: Analyse, design and create innovative products for addressing social needs.

PEO2: Equip themselves for employability, higher studies and research.

PEO3: Nurture the leadership qualities and entrepreneurial skills for their successful career.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO1:** Apply the basic and advanced knowledge in developing software, hardware and firmware solutions addressing real life problems.

- **PSO2:** Design, develop, test and implement product-based solutions for their career enhancement.

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

The Vehicle Parking Management System is a comprehensive and user-friendly solution designed to streamline the management of vehicle parking facilities. It addresses the growing demand for efficient utilization of limited parking spaces, automation of vehicle tracking, and accurate fee calculation. The system integrates key functionalities including real-time tracking of parking slot availability, automated entry and exit logging, duration-based fee calculation, and report generation for administrative use. At its core, the system maintains a relational database comprising vehicles, parking slots, and transaction records. Upon vehicle arrival, users can register their vehicle and be assigned an available slot. The system logs the entry time and marks the slot as occupied. When the vehicle exits, the exit time is recorded, and the system automatically calculates the parking fee based on the duration of stay using a configurable rate. The slot is then marked as available, updating the system in real time. To enhance management efficiency, the system provides detailed usage reports and payment summaries, helping administrators monitor peak usage periods, total earnings, and slot occupancy trends. A user-friendly interface ensures easy access and operation for both users and administrators.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD A DATABASE FOR REAL-TIME PROBLEMS

ABSTRACT	POs MAPPED	PSOs MAPPED
The Vehicle Parking Management System is a Java-based application developed using Java Swing for the GUI and MySQL for backend data management. This system aims to automate and streamline the management of vehicle parking in small to medium-sized facilities, replacing manual logbooks and improving operational efficiency. The system enables users to add new vehicles, assign available parking slots, track parking transactions, and automatically calculate parking charges based on the duration of stay. When a vehicle is added, it is immediately assigned to the next available slot, and a corresponding transaction is created. Upon vehicle exit, the system calculates the charge using a predefined rate (e.g., ₹20/hour) and updates the slot status for future use.	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -3** **PO5 -3** **PO6 -3** **PO7 -3** **PO8 -3** **PO9 -3** **PO10 -3** **PO11-3** **PO12 -3**	**PSO1 -3** **PSO2 -3**

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

DBMS	Database Management System
GUI	Graphical User Interface
SQL	Structured Query Language
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
RFID	Radio Frequency Identification
API	Application Programming Interface
XML	eXtensible Markup Language

CHAPTER 1
INTRODUCTION

1.1 Objective

With the rapid increase in the number of vehicles worldwide, managing parking spaces efficiently has become a significant challenge for urban areas, commercial complexes, and institutions. Traditional manual parking management systems are often inefficient, prone to errors, and result in poor user experience due to lack of real-time information and slow processing.

The Vehicle Parking Management System is designed to address these challenges by automating the entire parking process—from vehicle entry and slot allocation to exit and fee calculation. By leveraging technology, the system improves the utilization of parking spaces, reduces congestion, and ensures accurate tracking of vehicle movements.

1.2 Overview

The Vehicle Parking Management System is a web-based application designed to automate and streamline the management of parking facilities. It efficiently handles vehicle entries and exits, monitors parking slot availability in real-time, calculates fees based on the duration of parking, and generates detailed usage and payment reports. The system maintains records of vehicle details such as license plate numbers, types, and owner information, ensuring accurate tracking.

Parking slots are dynamically updated as "Available" or "Occupied" based on vehicle movement. Entry and exit times are logged automatically, and fees are calculated using predefined rates to eliminate manual errors. A user-friendly web interface developed using HTML, CSS, and PHP allows seamless interaction for both users and administrators. MySQL is used as the backend to securely store and manage all records. The system also provides administrative reports on daily transactions, revenue, and slot utilization, enabling better decision-making and operational efficiency. Suitable for various environments like malls, hospitals, and office complexes, this system enhances transparency, reduces manual workload, and significantly improves the overall parking experience.

This system is highly scalable and suitable for deployment in various locations such as shopping malls, hospitals, residential complexes, airports, schools, and office premises. By digitizing the parking process, it not only reduces administrative burden and human errors but also enhances transparency, improves space utilization, and significantly elevates the user experience.

1.3 Technologies Used

The Vehicle Parking Management System relies on a robust Database Management System (DBMS) to efficiently store, manage, and retrieve data related to vehicles, parking slots, transactions, and user interactions. The system uses MySQL, a relational DBMS, which supports structured data storage, query execution, and data integrity enforcement through relationships and constraints.

The core DBMS concepts implemented in the system include:

- **Tables and Relationships**: Data is organized into normalized tables such as Vehicles, ParkingSlots, and Transactions, each containing specific attributes. Foreign key relationships are used to connect these tables and maintain referential integrity. For example, VehicleID in the Transactions table references the Vehicles table to associate parking records with specific vehicles.
- **Primary and Foreign Keys**: Unique identification of records is ensured using Primary Keys (e.g., VehicleID, SlotID, TransactionID), while Foreign Keys maintain the logical relationships between different entities in the system.

- **Constraints and Validation**: The system uses constraints such as NOT NULL, UNIQUE, and CHECK to prevent invalid data entries. For instance, slot status can only be “Available” or “Occupied”.
- **Data Manipulation Language (DML)**: SQL queries are used to insert**,** update**,** delete, and select data. These queries are executed based on user or administrator actions through the PHP interface.
- **Transactions and Time Stamping**: The DBMS records timestamps for vehicle entry and exit, which are crucial for calculating parking duration and generating accurate billing. The use of CURRENT_TIMESTAMP ensures automatic and consistent time recording.

CHAPTER 2
PROJECT METHODOLOGY

2.1 Problem Description

In today's rapidly urbanizing world, the number of vehicles on roads is increasing exponentially, leading to significant challenges in managing parking spaces efficiently. Traditional parking systems often rely on manual processes, which are time-consuming, error-prone, and lack real-time monitoring capabilities. These systems fail to provide accurate information on slot availability, resulting in congestion, vehicle mismanagement, unnecessary delays, and user frustration.

Moreover, manual tracking of vehicle entry and exit times makes it difficult to calculate the exact duration of stay and corresponding parking fees, leading to revenue leakage and disputes. Administrators also face difficulties in maintaining proper records, generating reports, and analyzing parking usage trends for future planning. In large facilities such as shopping malls, hospitals, office complexes, and public parking areas, these issues are magnified due to high vehicle turnover.

2.2 Proposed Work

The proposed work involves the design and development of an automated Vehicle Parking Management System aimed at eliminating the inefficiencies and limitations of traditional parking methods. This web-based system will be built using Java Swing for server-side scripting, MySQL for database management, and HTML/CSS with optional JavaScript for creating an interactive and responsive user interface.

The system will automate critical processes such as vehicle registration, parking slot allocation, entry and exit time logging, and dynamic fee calculation based on parking duration. It will support real-time slot monitoring, ensuring that users and administrators can instantly view available or occupied slots.

To enhance accuracy, server-side timestamps will be used for recording entry and exit times, which will then be used to automatically calculate the parking fee based on customizable hourly or slab-based pricing models. The system will also feature payment status tracking and the ability to search, sort, and filter vehicle and transaction records. Administrators will have access to report generation tools to view daily, weekly, or monthly summaries of revenue, occupancy rates, and overall system usage. The system will include error handling, data validation, and user input checks to maintain data integrity.

It can also be configured for different user roles such as admin, operator, and customer, allowing role-based access to features. By implementing this system, parking facilities can significantly improve operational efficiency, reduce manual errors, enhance transparency, and offer a modern, user-centric experience.

2.3 ER Diagram

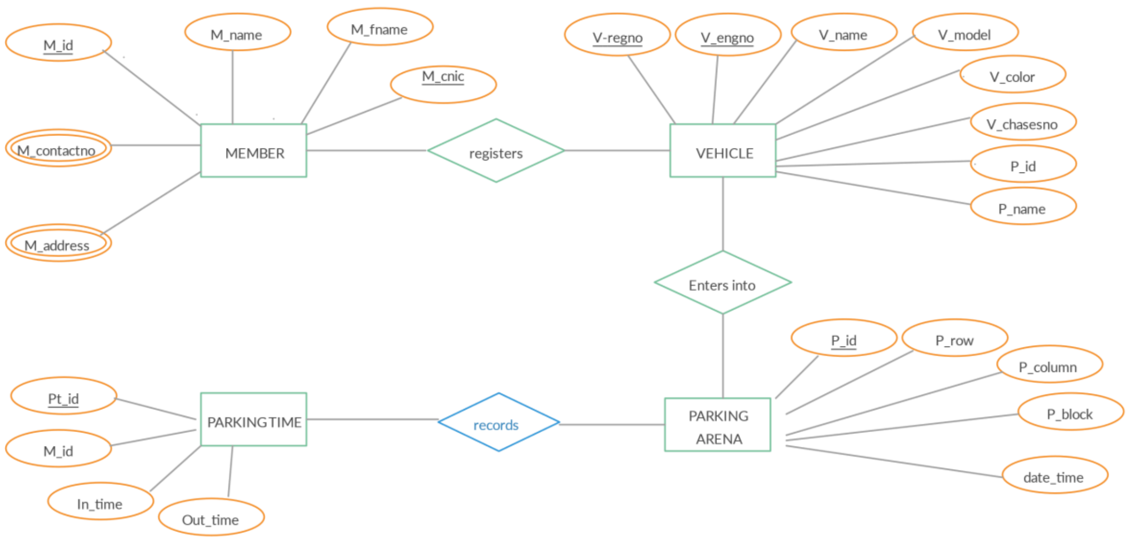


Fig 2.3.1 ER Diagram for Vehicle Management System

2.4 ER Diagram Overview

- MEMBER: Stores user details like M_id, M_name, M_contactno, etc.
- VEHICLE: Contains vehicle details such as V_regno, V_name, V_model, etc.
- PARKING ARENA: Represents parking slots with P_id, P_row, P_column, etc.
- PARKINGTIME: Tracks vehicle timing details like In_time, Out_time.
- A MEMBER registers one or more VEHICLEs.
- A VEHICLE enters into a PARKING ARENA.
- A PARKINGTIME record is linked to a MEMBER and stores entry/exit time.
- PARKINGTIME records are associated with PARKING ARENA entries.

CHAPTER 3
MODULE DESCRIPTION

3.1 Slots Module

The Slots Module manages the creation, display, and status tracking of all parking slots in the system. It displays a real-time view of all available and occupied slots by fetching data from the ParkingSlots table. Each slot is uniquely identified using a slot number or ID, and its status is visually represented on the webpage. When a vehicle is parked, the module updates the status to “Occupied” using an SQL UPDATE statement, and upon exit, the slot status is reset to “Available”. This module also prevents assigning already occupied slots by filtering them out from the dropdown during vehicle entry. Additional enhancements may include color-coded slot indicators, slot-type categorization (e.g., two-wheeler/four-wheeler), and future integration with IoT sensors for automated slot detection.

3.2 Add New Vehicle Module

The Add New Vehicle Module allows users or operators to register a vehicle into the system. It captures essential inputs such as the vehicle number, vehicle type, owner name, and selected parking slot. Upon submission, it inserts a new record into the Vehicles or ParkingRecord table along with the current timestamp (entry time), using JavaSwing and SQL. The selected slot is then marked as “Occupied” by invoking the Slots Module. Input validation is enforced to prevent blank or duplicate entries. This module ensures that only available slots can be selected and supports future enhancements like auto-suggestion for frequently visited vehicles, license plate verification, and printing of parking tokens or receipts.

3.3 Exit and Charge Module

The Exit and Charge Module is responsible for processing vehicle exits and calculating parking charges. When a vehicle is ready to leave, the system records the current time as the exit time and updates the respective transaction. The module calculates the duration by computing the time difference between the exit and entry timestamps, then applies a predefined rate (hourly or slab-based) to determine the total amount due. The charge is automatically displayed on the exit page for confirmation. It then updates the transaction record with the amount and exit time, and frees up the slot by updating its status to “Available”. Additional logic may include a minimum charge threshold, rounding of fees, or peak-time pricing. The module can be extended to include online/offline payment status and downloadable receipts.

3.4 View Transaction Module

The View Transaction Module provides a comprehensive view of all parking activities. It retrieves data from the ParkingRecord or Transactions table and displays fields such as transaction ID, vehicle number, slot ID, entry and exit times, duration, and calculated charge. For vehicles still parked, the system displays "Still Parked" in the exit time field and keeps the amount field blank or marked as pending. This module helps administrators monitor system activity in real-time, verify completed and ongoing transactions, and export data if needed. Additional features can include filters for date ranges, search by vehicle number or owner name, sorting by latest entry.

CHAPTER 4

IMPLEMENTATION & TEST CASES

4.1 TABLE STRUCTURES

Column	Data Type	Constraints
VehicleID	INT	PRIMARY KEY, AUTO_INCREMENT
LicensePlate	VARCHAR(15)	NOT NULL, UNIQUE
VehicleType	VARCHAR(20)	NOT NULL
OwnerName	VARCHAR(50)	NOT NULL

Table 4.1: Vehicles

Column	Data Type	Constraints
SlotID	INT	PRIMARY KEY, AUTO_INCREMENT
SlotNumber	VARCHAR(100	NOT NULL, UNIQUE
Status	VARCHAR(10)	CHECK (Status IN (‘Available’, ‘Occupied’))

Table 4.2: Parking Slots

Column	Data Type	Constraints
TransactionID	INT	PRIMARY KEY, AUTO_INCREMENT
VehicleID	INT	FOREIGN KEY → Vehicles(VehicleID)
SlotID	INT	FOREIGN KEY → ParkingSlots(SlotID)
EntryTime	DATETIME	DEFAULT CURRENT_TIMESTAMP
ExitTime	DATETIME	Nullable
ExitCharge	DECIMAL(10,2)	Nullable

Table 4.3: Transcations

4.2 SQL QUERIES

```
DROP DATABASE IF EXISTS ParkingManagement;
CREATE DATABASE ParkingManagement;
USE ParkingManagement;

CREATE TABLE Vehicles (
    VehicleID INT PRIMARY KEY AUTO_INCREMENT,
    LicensePlate VARCHAR(15) NOT NULL UNIQUE,
    VehicleType VARCHAR(20),
    OwnerName VARCHAR(50)
);
CREATE TABLE ParkingSlots (
    SlotID INT PRIMARY KEY AUTO_INCREMENT,
    SlotNumber VARCHAR(100) NOT NULL UNIQUE,
    Status VARCHAR(10) CHECK (Status IN ('Available', 'Occupied'))
);
CREATE TABLE Transactions (
    TransactionID INT PRIMARY KEY AUTO_INCREMENT,
    VehicleID INT,
    SlotID INT,
    EntryTime DATETIME DEFAULT CURRENT_TIMESTAMP,
    ExitTime DATETIME,
    ExitCharge DECIMAL(10,2),
    FOREIGN KEY (VehicleID) REFERENCES Vehicles(VehicleID),
    FOREIGN KEY (SlotID) REFERENCES ParkingSlots(SlotID)
);

INSERT INTO ParkingSlots (SlotNumber, Status) VALUES
('A', 'Available'),
('B', 'Available'),
('C', 'Available');

INSERT INTO Vehicles (LicensePlate, VehicleType, OwnerName) VALUES
('TN01AB1234', 'Car', 'John Doe'),
('TN02XY5678', 'Bike', 'Alice Smith');

INSERT INTO Transactions (VehicleID, SlotID, EntryTime) VALUES
(1, 1, NOW());
SELECT * FROM ParkingSlots WHERE Status = 'Available';
INSERT INTO Transactions (VehicleID, SlotID, EntryTime) VALUES (1, 1,
NOW());
UPDATE ParkingSlots SET Status = 'Occupied' WHERE SlotID = 1;
UPDATE Transactions SET ExitTime = NOW(), ExitCharge = 100 WHERE
TransactionID = 1;
UPDATE ParkingSlots SET Status = 'Available' WHERE SlotID = (SELECT
SlotID FROM Transactions WHERE TransactionID = 1);
SELECT
    VehicleID, SlotID, EntryTime,
    COALESCE(ExitTime, 'Still Parked') AS ExitTime,
    COALESCE(ExitCharge, 'Pending') AS ExitCharge
FROM Transactions;
```

4.3 TEST CASES & RESULTS

Test Case ID	Scenario	Description	Expected Result
TC01	Add a new vehicle	Add a vehicle with license plate TN03CD9999 belonging to Charlie Ray	New record in Vehicles with VehicleID = 3.
TC02	Check available slots	Retrieve all parking slots that are currently marked as 'Available'	Returns slots such as A, B, C (before any vehicle is parked).
TC03	Allocate slot to a vehicle	Assign Slot 2 to VehicleID = 3 and mark Slot 2 as 'Occupied'	New transaction is recorded; Slot 2 status changes to 'Occupied'.
TC04	Exit a vehicle and update charges	Set exit time and parking charge for a transaction, then mark the slot as 'Available'	Transaction updated with exit details; slot becomes available again.
TC05	View all transactions	Display all transactions, showing 'Still Parked' or 'Pending' for incomplete records	Table shows parking history with user-friendly labels for ongoing records.
TC06	Duplicate vehicle plate (error)	Attempt to insert a new vehicle with a license plate that already exists	Error: Duplicate entry violates unique constraint on LicensePlate.
TC07	Park in already occupied slot	Try to assign a slot that is already marked as 'Occupied' to a new transaction	Technically succeeds (unless restricted in app logic), but logically incorrect.
TC08	Invalid foreign key (VehicleID)	Try to create a transaction with a VehicleID that doesn't exist in the Vehicles table	Error: Foreign key constraint fails due to non-existent VehicleID.

Table 4.4: Test Cases & Results

CHAPTER 5
CONCLUSION & FUTURE SCOPE

5.1 CONCLUSION

The Vehicle Parking Management System successfully achieves its goal of providing a structured, automated, and efficient solution to manage vehicle entries, parking slot allocation, duration tracking, and fee calculation. It replaces traditional manual methods with a reliable web-based platform that reduces human errors, saves time, and enhances user experience.

By implementing real-time slot status updates, automatic timestamping, and dynamic billing, the system ensures accuracy and consistency across all operations. The modular structure—comprising vehicle entry, slot monitoring, transaction tracking, and reporting—makes the system highly maintainable and adaptable. Its user-friendly interface, secure database management, and clear administrative controls make it practical for everyday use in various public and private parking facilities such as malls, hospitals, residential complexes, and educational institutions. Overall, the system promotes transparency, improves parking efficiency, and supports better decision-making for administrators through accurate records and analytics.

The Vehicle Parking Management System not only automates and streamlines the entire parking process but also significantly enhances security by maintaining comprehensive logs of vehicle entries and exits, which aids in auditing and resolving disputes effectively. By reducing reliance on manual labor, the system lowers operational costs and minimizes human errors such as incorrect fee calculations or lost records. Transparent and timely billing fosters user trust and minimizes conflicts related to parking charges, improving overall customer satisfaction. The system's adaptable design allows customization for a variety of parking environments, from small office lots to large multi-level garages, and supports scalability to incorporate future features or integrations with traffic management and vehicle tracking systems.

5.2 FUTURE SCOPE

The future development of the Vehicle Parking Management System holds vast possibilities for enhancing functionality and user experience. Integration with IoT sensors could automate slot occupancy detection, eliminating manual status updates and increasing reliability. Implementation of RFID or QR code scanning could streamline vehicle entry and exit, reducing waiting times and minimizing human intervention.

Developing a mobile application would offer users the convenience of reserving parking spaces in advance, receiving real-time slot availability notifications, and making instant payments. Incorporating online payment gateways and digital wallets would improve transaction security and ease. Expanding the system to support multi-level and zoned parking structures can cater to large commercial complexes and urban settings. Adding AI-powered analytics could predict peak usage times and suggest dynamic pricing models to optimize revenue.

Features such as automated alerts for overstayed vehicles, integration with smart city infrastructure, and cloud-based data storage for scalability and disaster recovery will further future-proof the system. These advancements will enable the system to evolve into a fully automated, intelligent parking solution suitable for diverse, modern environments.

APPENDIX A
(SOURCE CODE)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import javax.swing.table.DefaultTableModel;

public class ParkingSystemApp extends JFrame {

    private static final String DB_URL =
"jdbc:mysql://localhost:3306/ParkingManagement";
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "Deni@2006";
    private Connection conn;

    public ParkingSystemApp() {
        connectDatabase();
        showIndexPage();
    }

    private void connectDatabase() {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            conn = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
            System.out.println("Connected to database.");
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "DB Connection Failed: " +
e.getMessage());
        }
    }

    private void showIndexPage() {
        setTitle("Vehicle Parking System - Home");
        setSize(420, 350);
        setLayout(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        getContentPane().setBackground(new Color(245, 245, 255)); // subtle light
purple

        Font font = new Font("Segoe UI", Font.PLAIN, 16);

        JButton viewSlotsBtn = createButton("View Slots", font);
        JButton addVehicleBtn = createButton("Add Vehicle", font);
        JButton viewTransBtn = createButton("View Transactions", font);
        JButton exitChargeBtn = createButton("Exit & Charge", font);

        viewSlotsBtn.setBounds(100, 40, 200, 40);
        addVehicleBtn.setBounds(100, 90, 200, 40);
        viewTransBtn.setBounds(100, 140, 200, 40);
        exitChargeBtn.setBounds(100, 190, 200, 40);

        add(viewSlotsBtn);
        add(addVehicleBtn);
        add(viewTransBtn);
        add(exitChargeBtn);

        viewSlotsBtn.addActionListener(e -> showViewSlotsPage());
        addVehicleBtn.addActionListener(e -> showAddVehiclePage());
        viewTransBtn.addActionListener(e -> showTransactionsPage());
        exitChargeBtn.addActionListener(e -> showExitChargePage());
    }

    private JButton createButton(String text, Font font) {
        JButton button = new JButton(text);
        button.setFont(font);
        button.setBackground(new Color(173, 216, 230)); // light blue
        button.setForeground(Color.BLACK);
        button.setFocusPainted(false);
        return button;
    }

    private void showViewSlotsPage() {
        JFrame frame = new JFrame("View Parking Slots");
        frame.setSize(450, 300);
        frame.setLocationRelativeTo(null);
        frame.getContentPane().setBackground(new Color(240, 248, 255)); // AliceBlue

        JTextArea area = new JTextArea();
        area.setEditable(false);
        area.setFont(new Font("Segoe UI", Font.PLAIN, 14));
        area.setBackground(new Color(255, 255, 240)); // Ivory
        JScrollPane scroll = new JScrollPane(area);
        frame.add(scroll);
```

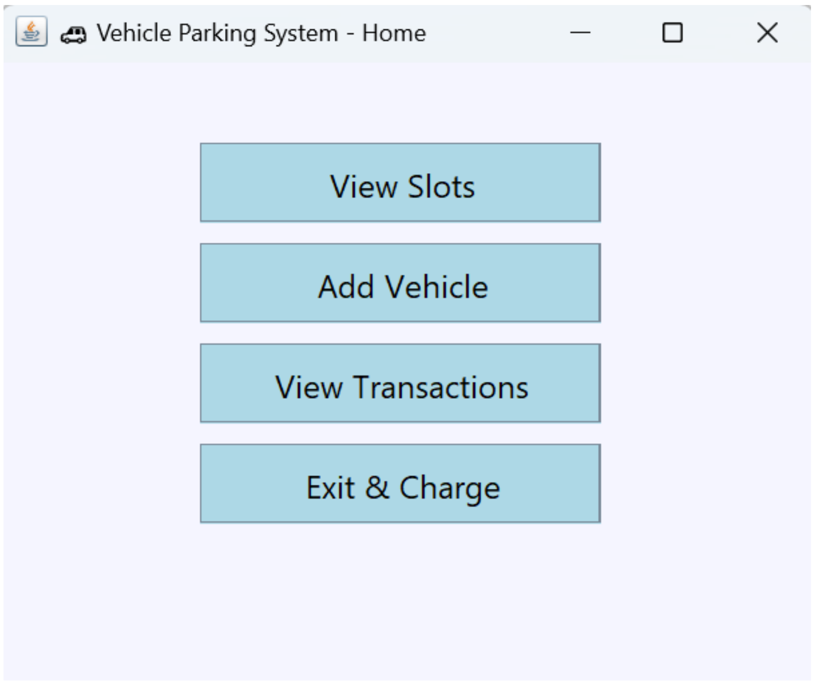
```
        try (Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery("SELECT * FROM ParkingSlots")) {
            while (rs.next()) {
                area.append("Slot: " + rs.getString("SlotNumber") + " - Status: " +
rs.getString("Status") + "\n");
            }
        } catch (SQLException e) {
            area.setText("Error loading slots: " + e.getMessage());
        }

        frame.setVisible(true);
    }

    private void showAddVehiclePage() {
        JFrame frame = new JFrame("Add New Vehicle");
        frame.setSize(400, 300);
        frame.setLocationRelativeTo(null);
        frame.getContentPane().setBackground(new Color(255, 250, 240)); //
FloralWhite
        frame.setLayout(new GridLayout(5, 2, 10, 10));

        JTextField licenseField = new JTextField();
        JTextField typeField = new JTextField();
        JTextField ownerField = new JTextField();
        JTextArea resultArea = new JTextArea();
        resultArea.setEditable(false);
        resultArea.setBackground(new Color(245, 255, 250)); // MintCream
        JButton addBtn = new JButton("Add Vehicle");
        addBtn.setBackground(new Color(144, 238, 144)); // LightGreen

        frame.add(new JLabel("License Plate:"));
        frame.add(licenseField);
        frame.add(new JLabel("Vehicle Type:"));
        frame.add(typeField);
        frame.add(new JLabel("Owner Name:"));
        frame.add(ownerField);
        frame.add(addBtn);
        frame.add(new JLabel());
        frame.add(new JScrollPane(resultArea));

        addBtn.addActionListener(e -> {
            try {
                // 1. Insert into Vehicles
                String sql = "INSERT INTO Vehicles (LicensePlate, VehicleType,
OwnerName) VALUES (?, ?, ?)";
                PreparedStatement ps = conn.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);
                ps.setString(1, licenseField.getText());
                ps.setString(2, typeField.getText());
                ps.setString(3, ownerField.getText());
                ps.executeUpdate();

                ResultSet keys = ps.getGeneratedKeys();
                if (keys.next()) {
                    int vehicleId = keys.getInt(1);
                // 2. Find an available slot
                Statement slotStmt = conn.createStatement();
                ResultSet slotRs = slotStmt.executeQuery("SELECT SlotID FROM
ParkingSlots WHERE Status = 'Available' LIMIT 1");

                if (slotRs.next()) {
                    int slotId = slotRs.getInt("SlotID");

                    // 3. Insert into Transactions
                    String txnSql = "INSERT INTO Transactions (VehicleID, SlotID,
EntryTime) VALUES (?, ?, NOW())";
                    PreparedStatement txnPs = conn.prepareStatement(txnSql);
                    txnPs.setInt(1, vehicleId);
                    txnPs.setInt(2, slotId);
                    txnPs.executeUpdate();

                    // 4. Update slot status
                    String updateSlot = "UPDATE ParkingSlots SET Status = 'Occupied'
WHERE SlotID = ?";
                    PreparedStatement slotUpdate = conn.prepareStatement(updateSlot);
                    slotUpdate.setInt(1, slotId);
                    slotUpdate.executeUpdate();

                    resultArea.setText("Vehicle Added\n Assigned to Slot ID: " + slotId);
                } else {
                    resultArea.setText("No available parking slots.");
                }
            }
            } catch (SQLException ex) {
                resultArea.setText("Error: " + ex.getMessage());
            }
        });

        frame.setVisible(true);
    }

    private void showTransactionsPage() {
        JFrame frame = new JFrame("View Transactions");
        frame.setSize(700, 300);
        frame.setLocationRelativeTo(null);
        frame.getContentPane().setBackground(new Color(240, 255, 255)); // Azure

        String[] columnNames = {"Transaction ID", "Vehicle ID", "Slot ID", "Entry
Time", "Exit Time", "Charge"};
        DefaultTableModel tableModel = new DefaultTableModel(columnNames, 0);
        JTable table = new JTable(tableModel);
        table.setFillsViewportHeight(true);
        table.setFont(new Font("Segoe UI", Font.PLAIN, 14));
        table.setRowHeight(25);
        table.getTableHeader().setFont(new Font("Segoe UI", Font.BOLD, 14));
        table.setBackground(new Color(255, 248, 220)); // Cornsilk

        JScrollPane scrollPane = new JScrollPane(table);
        frame.add(scrollPane);

        try (Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery("SELECT * FROM Transactions")) {
```

```
            while (rs.next()) {
                int tid = rs.getInt("TransactionID");
                int vid = rs.getInt("VehicleID");
                int sid = rs.getInt("SlotID");
                String entry = rs.getString("EntryTime");
                String exit = rs.getString("ExitTime") != null ? rs.getString("ExitTime") :
"Still Parked";
                String charge = rs.getString("ExitCharge") != null ? "₹" +
rs.getString("ExitCharge") : "Pending";

                Object[] rowData = {tid, vid, sid, entry, exit, charge};
                tableModel.addRow(rowData);
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(frame, "Error loading transactions: " +
e.getMessage());
        }

        frame.setVisible(true);
    }

    private void showExitChargePage() {
        JFrame frame = new JFrame("Exit and Charge");
        frame.setSize(400, 250);
        frame.setLocationRelativeTo(null);
        frame.getContentPane().setBackground(new Color(255, 245, 238));
        frame.setLayout(new GridLayout(4, 2, 10, 10));

        JTextField transactionIdField = new JTextField();
        JTextArea resultArea = new JTextArea();
        resultArea.setEditable(false);
        resultArea.setBackground(new Color(255, 255, 224));

        JButton updateBtn = new JButton("Calculate & Update");
        updateBtn.setBackground(new Color(255, 182, 193));

        frame.add(new JLabel("Transaction ID:"));
        frame.add(transactionIdField);
        frame.add(updateBtn);
        frame.add(new JLabel());
        frame.add(new JScrollPane(resultArea));

        updateBtn.addActionListener(e -> {
            try {
                int transactionId = Integer.parseInt(transactionIdField.getText());

                // Get EntryTime
                String getSql = "SELECT EntryTime FROM Transactions WHERE
TransactionID = ?";
                PreparedStatement getStmt = conn.prepareStatement(getSql);
                getStmt.setInt(1, transactionId);
                ResultSet rs = getStmt.executeQuery();

                if (rs.next()) {
                    Timestamp entryTime = rs.getTimestamp("EntryTime");
                    Timestamp exitTime = new Timestamp(System.currentTimeMillis());

                    long diffMillis = exitTime.getTime() - entryTime.getTime();
                    long minutes = diffMillis / (1000 * 60);
                    long hours = (long) Math.ceil(minutes / 60.0);
                    double charge = hours * 20.0;

                    // Update DB with calculated charge
                    String updateSql = "UPDATE Transactions SET ExitTime = ?,
ExitCharge = ? WHERE TransactionID = ?";
                    PreparedStatement updateStmt = conn.prepareStatement(updateSql);
                    updateStmt.setTimestamp(1, exitTime);
                    updateStmt.setDouble(2, charge);
                    updateStmt.setInt(3, transactionId);
                    int updated = updateStmt.executeUpdate();

                    if (updated > 0) {
                        resultArea.setText("Exit Time: " + exitTime +
                                "\nDuration: " + minutes + " minutes" +
                                "\nCharge: ₹" + charge);
                    } else {
                        resultArea.setText("Transaction not found.");
                    }

                } else {
                    resultArea.setText("No transaction with that ID.");
                }

            } catch (Exception ex) {
                resultArea.setText("Error: " + ex.getMessage());
            }
        });
```

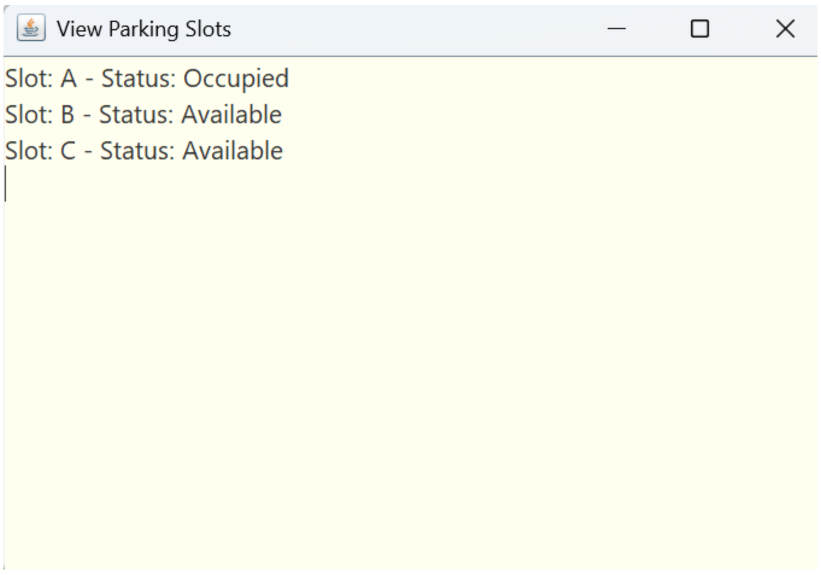
```
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            new ParkingSystemApp().setVisible(true);
        });
    }
}
```

APPENDIX B
(SCREENSHOTS)

Vehicle Parking System- Welcome Window

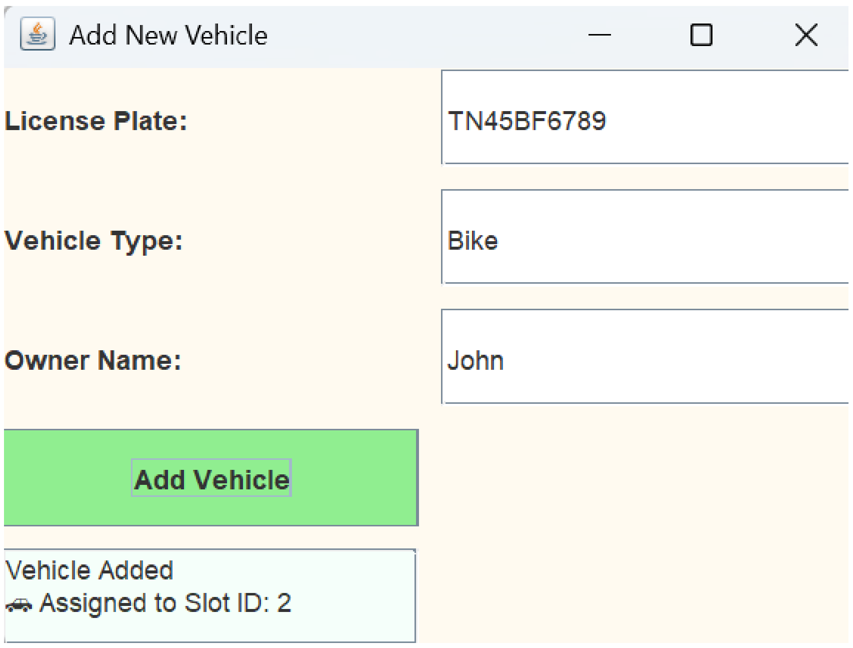


Vehicle Parking System – Parking Slots Window



```
Slot: A - Status: Occupied
Slot: B - Status: Available
Slot: C - Status: Available
```

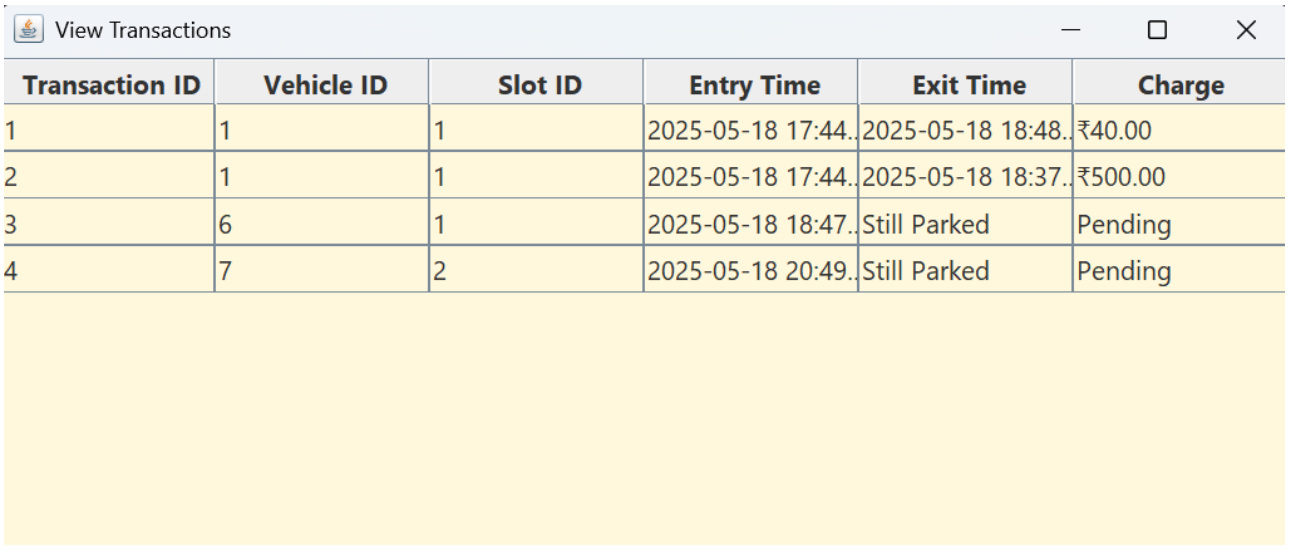
Vehicle Parking System - Add New Vehicle Window



License Plate:	TN45BF6789
Vehicle Type:	Bike
Owner Name:	John

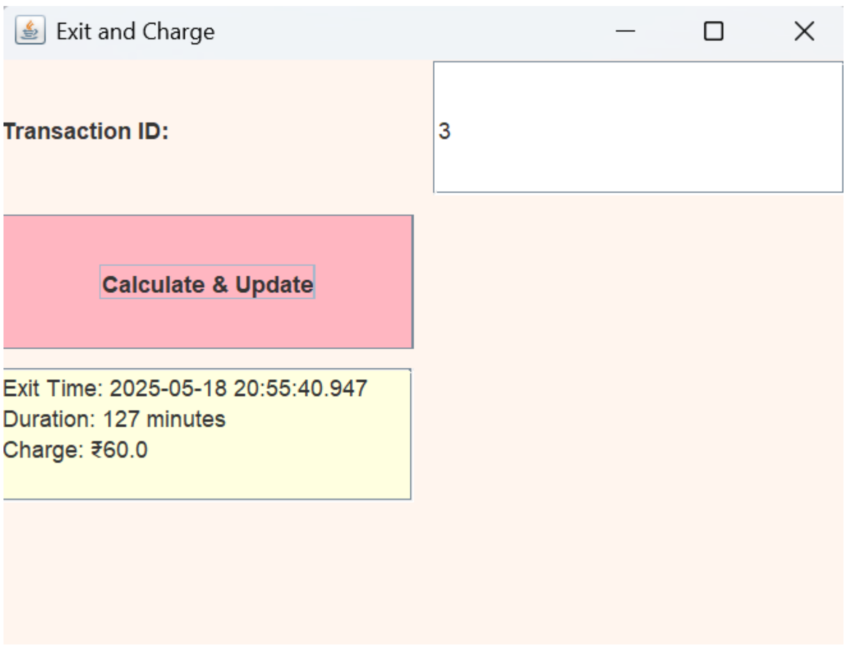
Vehicle Added
🚗 Assigned to Slot ID: 2

Vehicle Parking System – View Transactions Window



Transaction ID	Vehicle ID	Slot ID	Entry Time	Exit Time	Charge
1	1	1	2025-05-18 17:44..	2025-05-18 18:48..	₹40.00
2	1	1	2025-05-18 17:44..	2025-05-18 18:37..	₹500.00
3	6	1	2025-05-18 18:47..	Still Parked	Pending
4	7	2	2025-05-18 20:49..	Still Parked	Pending

Vehicle Parking System – Exit and Charge Window



Transaction ID: 3

```
Exit Time: 2025-05-18 20:55:40.947
Duration: 127 minutes
Charge: ₹60.0
```

REFERENCES

1. Herbert Schildt. “Java: The Complete Reference”, 11th Edition. McGraw Hill Education, 2018.
2. https://docs.oracle.com/javase/tutorial/uiswing/
3. https://www.tutorialspoint.com/swing/index.htm