

SMART PARKING SYSTEM



A PROJECT REPORT

Submitted by

VENKATACHALAM N(2303811724321120)

in partial fulfillment of requirements for the award of the course

CGB1201 – JAVA PROGRAMMING

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by
AICTE, New Delhi)


SAMAYAPURAM – 621 112

DECEMBER, 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)
SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on “SMART PARKING SYSTEM
” is the bonafide work of **VENKATACHALAM N 2303811724321120** who carried out
the project work during the academic year 2024 - 2025 under my supervision.



Signature

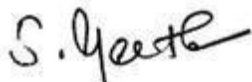
Dr. T. AVUDAIAPPAN M.E., Ph.D.,
HEAD OF THE DEPARTMENT,
Department of Artificial Intelligence,
K. Ramakrishnan College of Technology,
Samayapuram, Trichy -621 112.



Signature

Mrs. S. GEETHA M.E.,
SUPERVISOR,
Department of Artificial Intelligence,
K. Ramakrishnan College of Technology,
Samayapuram, Trichy -621 112.

Submitted for the viva-voce examination held on 3.12.24 .



INTERNAL EXAMINER



EXTERNAL EXAMINER

DECLARATION

I declare that the project report on “**SMART PARKING SYSTEM MANAGEMENT**” is the result of original work done by me and best of my knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This project report is submitted on the partial fulfillment of the requirement of the award of the **CGB1201 – JAVA PROGRAMMING**.



Signature

VENKATACHALAM N

Place: Samayapuram

Date: 3/12/2024

ACKNOWLEDGEMENT

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

I extend our sincere acknowledgment and appreciation to the esteemed and honourable Chairman, **Dr. K. RAMAKRISHNAN, B.E.**, for having provided 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 our project and offering an adequate duration to complete it.

I would like to thank **Dr. N. VASUDEVAN, M.TECH., Ph.D.**, Principal, who gave the opportunity to frame the project to full satisfaction.

I thank **Dr. T. AVUDAIAPPAN, M.E., Ph.D.**, Head of the Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for providing her encouragement in pursuing this project.

I wish to convey our profound and heartfelt gratitude to our esteemed project guide **Mrs. S. GEETHA M.E.**, Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for her incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project.

I render our sincere thanks to the 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 serve the society by offering top-notch technical education on par with global standards.

MISSION OF THE INSTITUTION

- Be a centre of excellence for technical education in emerging technologies by exceeding the needs of industry and society.
- Be an institute with world class research facilities.
- Be an institute nurturing talent and enhancing competency of students to transform them as all- round personalities respecting moral and ethical values.

VISION AND MISSION OF THE DEPARTMENT

To excel in education, innovation and research in Artificial Intelligence and Data Science to fulfill industrial demands and societal expectations.

Mission 1: To educate future engineers with solid fundamentals, continually improving teaching methods using modern tools.

Mission 2: To collaborate with industry and offer top-notch facilities in a conducive learning environment.

Mission 3: To foster skilled engineers and ethical innovation in AI and Data Science for global recognition and impactful research.

Mission 4: To tackle the societal challenge of producing capable professionals by instilling employability skills and human values.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO 1: Compete on a global scale for a professional career in Artificial Intelligence and Data Science.

PEO 2: Provide industry-specific solutions for the society with effective communication and ethics.

PEO 3: Hone their professional skills through research and lifelong learning initiatives.

PROGRAM OUTCOMES

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.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO 1:** Capable of working on data-related methodologies and providing industry-focussed solutions.
- **PSO2:** Capable of analysing and providing a solution to a given real-world problem by designing an effective program.

ABSTRACT

The **Smart Parking System** is a Java-based solution aimed at solving the growing problem of parking space management in urban areas. The system uses real-time data, sensors, and automation to monitor parking space availability, providing drivers with up-to-date information on vacant spots. By integrating an easy-to-use interface, the system allows users to search for, reserve, and pay for parking spaces via a mobile or web application. This reduces the time spent searching for parking and helps alleviate traffic congestion caused by drivers circulating through crowded parking areas.

In addition to user convenience, the system provides parking lot administrators with powerful tools to manage parking spaces, track reservations, and monitor usage patterns. The project incorporates Java technologies such as JDBC for database connectivity, real-time data management, and a graphical user interface (GUI) built with JavaFX or Swing. It also includes payment integration, offering secure and efficient online payment options. With its ability to collect and analyze parking data, the Smart Parking System optimizes space utilization and contributes to more efficient urban mobility.

.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No.
	ABSTRACT	viii
1	INTRODUCTION	1
	1.1 INTRODUCTION	1
	1.2 OBJECTIVE	2
2	PROJECT METHODOLOGY	4
	2.1 PROPOSED WORK	4
	2.2 BLOCK DIAGRAM	5
3	JAVA PROGRAMMING CONCEPTS	6
	3.1 OBJECT ORIENTED PROGRAMMING (OOP)	6
	3.2 CONCURRENCY AND DATABASEMANAGEMENT	7
4	MODULE DESCRIPTION	8
	4.1 USER AUTHENTICATION	8
	4.2 PARKING SPACE MANAGEMENT	8
	4.3 RESERVATION AND BOOKING	9
	4.4 PAYMENT AND FILLING	9
	4.5 REPORTING MODULE	10
5	CONCLUSION	11
	REFERENCES	12
	APPENDICES	13
	Appendix A – Source code	13
	Appendix B – Screen shots	22

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The Smart Parking System is an innovative solution designed to optimize the process of parking in urban spaces, especially in crowded cities. It aims to make the parking experience more efficient, user-friendly, and automated by integrating technology. With the growing number of vehicles and limited parking spaces, finding a free spot can often be a time-consuming and frustrating task. The Smart Parking System addresses this issue by providing real-time information about parking space availability, guiding users to the nearest available spot, and automating the parking process.

In this project, a Smart Parking System is developed using Java, which is a widely used programming language known for its portability, scalability, and robustness. The system aims to:

- **Track Parking Space Availability:** The system monitors the occupancy of parking spaces and provides users with real-time updates about available spaces.
- **User-Friendly Interface:** A user-friendly interface for both the driver and the parking manager, enabling easy interaction with the system.
- **Efficient Parking Management:** The parking lot management system automates several tasks, such as registering vehicles, calculating parking fees, and ensuring optimal space utilization.
- **Automated Payment System:** Integration with payment gateways to allow for automated payments when users park their vehicles.

1.2 OBJECTIVE

The objective of the **Smart Parking System** project is to develop an efficient and automated solution for managing parking spaces in busy urban environments. By leveraging real-time data, the system aims to help drivers easily find available parking spots, reducing the time spent searching for parking and minimizing traffic congestion. The project also seeks to automate various parking processes, such as vehicle entry and exit tracking, reservation management, and payment processing. By providing a streamlined experience, the system enhances user convenience, allowing drivers to reserve spaces in advance, navigate directly to available spots, and make payments electronically.

In addition to benefiting users, the system aims to improve parking lot management by offering tools to monitor and manage parking space occupancy. Admins can track space availability, adjust pricing dynamically, and generate usage reports to optimize operations and revenue. The system's design focuses on scalability, allowing it to be implemented in a variety of settings, from small private lots to large public parking facilities. Ultimately, the project strives to enhance the overall efficiency, convenience, and profitability of parking systems through automation and advanced technology integration.

The **Smart Parking System** project aims to leverage technology to address the growing challenges associated with urban parking. By integrating IoT sensors and real-time data processing, the system can monitor and manage parking spaces efficiently. This solution not only provides convenience to users but also reduces the environmental impact caused by unnecessary driving in search of parking spots. Through automated management and data analysis, the system can optimize the use of available spaces, ensuring maximum occupancy and better space utilization.

The project also focuses on enhancing the user experience by providing a simple and intuitive interface. Users can access the system through various platforms, such as mobile apps or web browsers, to check parking availability, make reservations, and complete payments. Furthermore, the system can support various types of users, including individual drivers and businesses, offering tailored features such as subscription-based services or daily parking options. This versatility makes the system adaptable to a wide range of environments, from city streets to corporate parking lots, contributing to a smarter, more sustainable approach to urban mobility.

CHAPTER 2

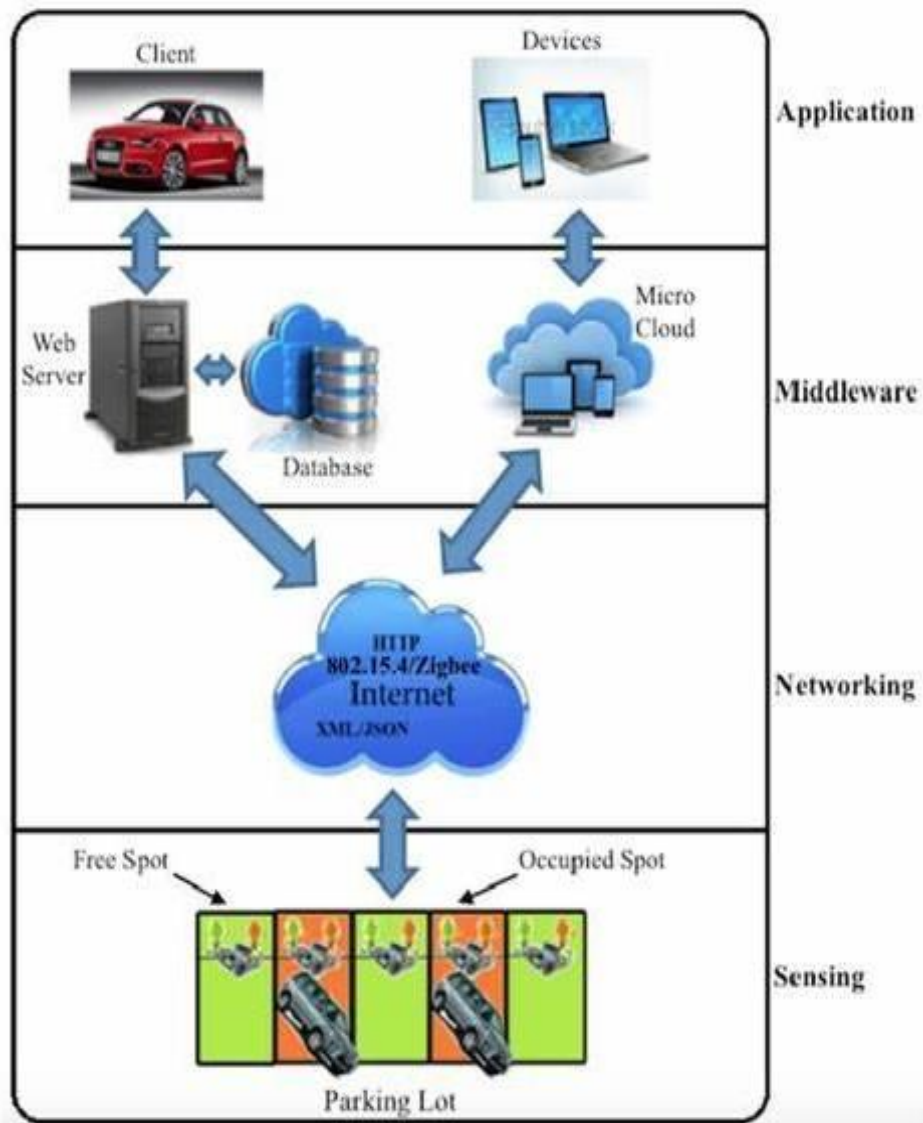
PROJECT METHODOLOGY

2.1 PROPOSED WORK

The **Smart Parking System** will be developed as a comprehensive solution to manage parking spaces in real-time, offering convenience to users and enhancing operational efficiency for parking lot managers. The proposed work involves the design and implementation of the system in several phases, which include system architecture, database management, user interface design, and integration with hardware (IoT sensors). Below are the key components of the proposed work

1. System Architecture and design
2. Real time Parking Space Availability
3. User Interface and User Experience
4. Parking Reservation and management

2.2 BLOCK DIAGRAM



CHAPTER 3

JAVA PROGRAMMING CONCEPTS

3.1 OBJECT ORIENTED PROGRAMMING(OOP)

The Smart Parking System relies heavily on Object-Oriented Programming (OOP) principles to structure the system in a modular, maintainable, and scalable manner. Java, being an object-oriented language, allows the system to be broken down into objects representing real-world entities and actions.

- **Classes and Objects:** In the Smart Parking System, various real-world entities like parking spaces, users, and reservations are represented as objects in the system. For example, the `ParkingSpace` class might have attributes like `spaceId`, `isOccupied`, and methods such as `reserveSpace()` or `freeSpace()`. These attributes and behaviors are encapsulated within the class and accessed through objects created from these classes.
- **Encapsulation:** By encapsulating the data (e.g., parking space status or user information) within classes and restricting access through getter and setter methods, the system ensures that data is protected from direct modification, maintaining consistency and reliability.
- **Inheritance and Polymorphism:** Java's inheritance allows for extending classes, enabling the creation of specialized entities, such as `PremiumUser`, which could extend the general `User` class. Polymorphism is applied when different types of users or parking spaces may require different methods of calculating fees or reserving spots, allowing the system to be more flexible and adaptable.

These OOP concepts help in organizing the system's components logically, making the development process easier, and allowing for easier updates and maintenance as the system grows or changes.

3.2 CONCURRENCY AND DATABASEMANAGEMENT

The Smart Parking System also needs to handle multiple user interactions simultaneously and manage large amounts of data effectively. This is where concepts like Concurrency and Database Connectivity come into play.

- **Concurrency and Multithreading:** Given that multiple users can simultaneously interact with the system to check parking space availability, make reservations, or process payments, multithreading is essential to ensure smooth performance. For example, a thread could be dedicated to monitoring and updating parking space availability in real time, while another thread handles user reservations and payment processing. This parallel processing prevents the system from slowing down when handling high traffic and multiple requests at once.
- **Database Connectivity (JDBC):** The Smart Parking System needs to store and retrieve user information, parking space data, reservations, and payment details. Java's JDBC (Java Database Connectivity) allows the system to interact with a relational database, such as MySQL or SQLite, to perform operations like adding a reservation, checking space availability, and storing transaction details. SQL queries can be executed from Java, and the data can be stored persistently for future use, such as retrieving previous parking history or calculating parking fees.

These concepts ensure that the Smart Parking System is not only able to handle multiple tasks simultaneously but also able to manage large datasets efficiently, providing a seamless experience for users and administrators.

CHAPTER 4

MODULE DESCRIPTION

4.1. USER AUTHENTICATION

Description:

This module handles user registration, login, and account management for both regular users and administrators. It allows users to create accounts, log in, update personal information, and manage their preferences, such as notification settings and payment methods. For administrators, the module provides a secure login system with privileged access to manage parking spaces, view reports, and configure system settings.

- Key Features:
 - User registration and login
 - Profile management (e.g., username, password, payment methods)
 - Admin panel for managing users and parking lot operations
 - User authentication with password encryption

4.2 PARKING SPACE MANAGEMENT

Description:

This module manages the availability and occupancy status of parking spaces in real time. It uses sensors (or a simulated system) to monitor whether parking spots are occupied or vacant. The system provides users with real-time information on available parking spaces and allows administrators to update the status of spaces, allocate new spaces, or remove outdated ones.

- Key Features:
 - Real-time tracking of parking space occupancy
 - Parking space availability display for users
 - Admin interface to add, update, or remove parking spaces
 - Integration with sensors or simulation of space occupancy

4.3 RESERVATION AND BOOKING

Description:

This module allows users to reserve parking spaces in advance, either by selecting available spaces from a map or by setting specific dates and times. The system checks for space availability in real time and confirms reservations. It also provides features for canceling or modifying reservations, ensuring that users have an efficient booking experience.

- Key Features:
 - User interface for selecting and reserving available spaces
 - Reservation confirmation and cancellation features
 - Real-time updates on availability
 - Reservation history tracking

4.4 PAYMENT AND BILLING

Description:

The payment and billing module is responsible for processing user payments when reserving parking spaces or using parking services. It supports multiple payment methods, including credit/debit cards, mobile payments, and possibly subscription-based billing. The module calculates parking fees based on factors like duration, space type, and time of day, and integrates with payment gateways for seamless transactions.

- Key Features:
 - Integration with payment gateways (e.g., Stripe, PayPal)
 - Fee calculation based on duration and space type
 - Digital payment processing (credit cards, mobile apps)
 - Receipt generation and transaction history for users

4.5 REPORTING MODULE

Description:

This module provides a comprehensive dashboard for parking lot administrators to monitor the system's performance. It includes features for managing parking space allocation, tracking reservations, monitoring user activity, and generating reports on parking usage, revenue, and payment transactions. The admin dashboard helps optimize the operation of the parking lot, allowing for real-time decision-making.

- Key Features:
 - Real-time monitoring of parking space usage and reservations
 - Revenue tracking and transaction history
 - Reporting tools for parking lot performance and usage analytics
 - Ability to adjust pricing and manage parking space availability

Each of these modules works together to create a complete and functional Smart Parking System, ensuring a seamless and efficient experience for users and administrators alike.

CHAPTER 5

CONCLUSION

The **Smart Parking System** is an innovative solution designed to address the growing challenges of parking in urban areas. By leveraging advanced technologies such as real-time data collection, sensor integration, user authentication, and automated payment systems, the project aims to enhance the efficiency and convenience of parking management. The system not only benefits users by providing an easy-to-use interface for finding and reserving parking spots but also streamlines operations for parking lot administrators with tools for monitoring space usage, managing reservations, and generating performance reports.

By implementing key Java programming concepts such as Object-Oriented Programming (OOP), Concurrency, and JDBC for database management, the Smart Parking System ensures scalability, reliability, and real-time functionality. These features contribute to an optimized, user-friendly experience, making the system adaptable to a wide range of parking environments—from shopping malls to urban parking lots.

As urban areas continue to grow, the demand for smarter, more efficient parking solutions will increase, making the Smart Parking System an essential tool for improving mobility and reducing traffic congestion. With the ability to expand and integrate with IoT devices, this system presents a forward-thinking approach to urban parking challenges.

REFERENCES:

Books:

1. "Java: The Complete Reference" by Herbert Schildt
 - A comprehensive guide to Java, covering key concepts for building systems like a Smart Parking System.
 - Reference: Schildt, H. (2018). *Java: The Complete Reference*.
2. "Head First Java" by Kathy Sierra & Bert Bates
 - A beginner-friendly book that explains Java concepts in an engaging way.
 - Reference: Sierra, K., & Bates, B. (2005). *Head First Java*.
3. "Java Programming: From Problem Analysis to Program Design" by D.S. Malik
 - A detailed resource for Java programming and design, ideal for system development.
 - Reference: Malik, D.S. (2012). *Java Programming*.
4. "Java Concurrency in Practice" by Brian Goetz
 - Focuses on Java's concurrency features, essential for handling multiple tasks in the Smart Parking System.
 - Reference: Goetz, B. (2006). *Java Concurrency in Practice*.

Websites:

1. Oracle Java Documentation
 - The official Java reference for classes, methods, and APIs.
 - Website: <https://docs.oracle.com/en/java/>
2. Java2s
 - A site with Java tutorials and code examples.
 - Website: <https://www.java2s.com/>
3. GeeksforGeeks - Java
 - Tutorials and explanations for various Java concepts.

- Website: <https://www.geeksforgeeks.org/java/>

4. W3Schools Java Tutorial

- A beginner-friendly Java tutorial with interactive examples.
- Website: <https://www.w3schools.com/java/>

APPENDICES

APPENDIX A – SOURCE CODE

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.time.LocalTime;
import java.util.HashMap;
import java.util.Map;

class ParkingSpace {
    String id;    // Unique ID of the parking space
    double price; // Base price of the parking space
    boolean reserved; // Whether the space is reserved
    boolean isVIP; // Whether the space is VIP

    public ParkingSpace(String id, double price, boolean isVIP) {
        this.id = id;
        this.price = price;
        this.reserved = false;
        this.isVIP = isVIP;
    }

    public double getDynamicPrice() {
        int hour = LocalTime.now().getHour();
        if (hour >= 8 && hour <= 18) { // Peak hours: 8 AM to 6 PM
```

```

        return price * 1.2;
    } else { // Off-peak hours
        return price * 0.9;
    }
}

```

@Override

```

public String toString() {
    return "Space ID: " + id + ", Price: $" + String.format("%.2f", getDynamicPrice())
+
        " | " + (isVIP ? "VIP" : "Regular");
}
}

```

```

class User {
    String id;
    Map<String, ParkingSpace> reservationHistory; // History of reserved spaces

```

```

    public User(String id) {
        this.id = id;
        this.reservationHistory = new HashMap<>();
    }

```

```

    public void viewReservationHistory(JTextArea historyArea) {
        if (reservationHistory.isEmpty()) {
            historyArea.setText("No reservations made yet.");
        } else {
            StringBuilder history = new StringBuilder("Your reservation history:\n");
            for (ParkingSpace space : reservationHistory.values()) {

```



```

        history.append(space).append("\n");
    }
    historyArea.setText(history.toString());
}
}

```

```

public boolean reserveParkingSpace(ParkingSpace space) {
    if (!space.reserved) {
        space.reserved = true;
        reservationHistory.put(space.id, space);
        return true;
    }
    return false;
}

```

```

public void vacateParkingSpace(ParkingSpace space) {
    space.reserved = false;
    reservationHistory.remove(space.id);
}
}

```

```

public class ParkingSystemApp {
    private JFrame frame;
    private Map<String, ParkingSpace> parkingSpaces = new HashMap<>();
    private Map<String, User> users = new HashMap<>();
    private User currentUser;

    public ParkingSystemApp() {
        initializeParkingSpaces();
    }
}

```

```

    initializeUsers();
    initializeGUI();
}

private void initializeParkingSpaces() {
    parkingSpaces.put("A1", new ParkingSpace("A1", 5.0, false));
    parkingSpaces.put("A2", new ParkingSpace("A2", 7.0, true));
    parkingSpaces.put("A3", new ParkingSpace("A3", 6.0, false));
    parkingSpaces.put("B1", new ParkingSpace("B1", 4.0, false));
    parkingSpaces.put("B2", new ParkingSpace("B2", 8.0, true));
    parkingSpaces.put("B3", new ParkingSpace("B3", 6.5, false));
}

private void initializeUsers() {
    users.put("user1", new User("user1"));
    users.put("user2", new User("user2"));
}

private void initializeGUI() {
    frame = new JFrame("Parking System");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(600, 400);
    frame.setLayout(new BorderLayout());

    JPanel userPanel = new JPanel();
    JLabel userLabel = new JLabel("Enter Username:");
    JTextField userField = new JTextField(10);
    JButton loginButton = new JButton("Login");
    userPanel.add(userLabel);

```

```

userPanel.add(userField);
userPanel.add(loginButton);

JPanel actionPanel = new JPanel();
JButton reserveButton = new JButton("Reserve");
JButton vacateButton = new JButton("Vacate");
JButton historyButton = new JButton("View History");
JTextArea infoArea = new JTextArea(10, 50);
infoArea.setEditable(false);
JScrollPane scrollPane = new JScrollPane(infoArea);

actionPanel.add(reserveButton);
actionPanel.add(vacateButton);
actionPanel.add(historyButton);

frame.add(userPanel, BorderLayout.NORTH);
frame.add(scrollPane, BorderLayout.CENTER);
frame.add(actionPanel, BorderLayout.SOUTH);

// Login Button Action
loginButton.addActionListener(e -> {
    String username = userField.getText().trim().toLowerCase();    // Normalize
input
    currentUser = users.get(username);
    if (currentUser != null) {
        infoArea.setText("Welcome, " + username + "!\n");
        displayParkingSpaces(infoArea);
    } else {
        infoArea.setText("Invalid username. Available usernames: user1, user2.\n");
    }
});

```

```

    }
});

// Reserve Button Action
reserveButton.addActionListener(e -> {
    if (currentUser == null) {
        infoArea.setText("Please log in first.\n");
        return;
    }
    String spaceId = JOptionPane.showInputDialog(frame, "Enter Space ID to
Reserve:").toUpperCase();
    ParkingSpace space = parkingSpaces.get(spaceId);
    if (space == null) {
        infoArea.setText("Space " + spaceId + " does not exist.\n");
    } else if (space.reserved) {
        infoArea.setText("Space " + spaceId + " is already reserved.\n");
    } else {
        if (currentUser.reserveParkingSpace(space)) {
            infoArea.setText("Space " + spaceId + " has been reserved.\n");
            displayParkingSpaces(infoArea);
        } else {
            infoArea.setText("Unable to reserve the space. Please try again.\n");
        }
    }
});

// Vacate Button Action
vacateButton.addActionListener(e -> {
    if (currentUser == null) {

```

```

        infoArea.setText("Please log in first.\n");
        return;
    }
    String spaceId = JOptionPane.showInputDialog(frame, "Enter Space ID to
Vacate:").toUpperCase();
    ParkingSpace space = parkingSpaces.get(spaceId);
    if (space == null) {
        infoArea.setText("Space " + spaceId + " does not exist.\n");
    } else if (!space.reserved) {
        infoArea.setText("Space " + spaceId + " is not reserved.\n");
    } else {
        currentUser.vacateParkingSpace(space);
        infoArea.setText("Space " + spaceId + " has been vacated.\n");
        displayParkingSpaces(infoArea);
    }
});

// History Button Action
historyButton.addActionListener(e -> {
    if (currentUser == null) {
        infoArea.setText("Please log in first.\n");
        return;
    }
    currentUser.viewReservationHistory(infoArea);
});

frame.setVisible(true);
}

```

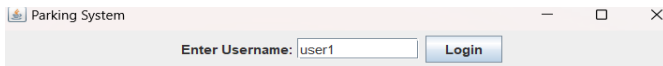
```

private void displayParkingSpaces(JTextArea infoArea) {
    StringBuilder display = new StringBuilder("Available Parking Spaces:\n");
    for (ParkingSpace space : parkingSpaces.values()) {
        String status = space.reserved ? "Reserved" : "Available";
        display.append(space).append(" | Status: ").append(status).append("\n");
    }
    infoArea.setText(display.toString());
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(ParkingSystemApp::new);
}
}

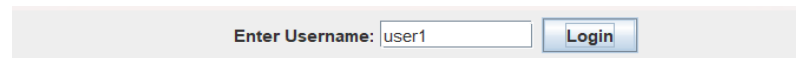
```

APPENDIX B – SCREENSHOTS



Parking System

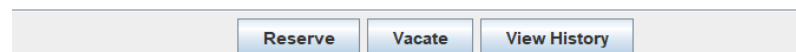
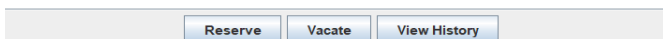
Enter Username:



Enter Username:

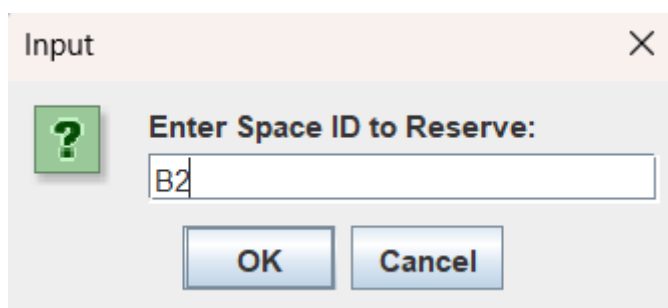
Available Parking Spaces:

- Space ID: A1, Price: \$4.50 | Regular | Status: Available
- Space ID: B2, Price: \$7.20 | VIP | Status: Available
- Space ID: A2, Price: \$6.30 | VIP | Status: Available
- Space ID: B3, Price: \$5.85 | Regular | Status: Available
- Space ID: A3, Price: \$5.40 | Regular | Status: Available
- Space ID: B1, Price: \$3.60 | Regular | Status: Available



Available Parking Spaces:

- Space ID: A1, Price: \$4.50 | Regular | Status: Available
- Space ID: B2, Price: \$7.20 | VIP | Status: Reserved
- Space ID: A2, Price: \$6.30 | VIP | Status: Available
- Space ID: B3, Price: \$5.85 | Regular | Status: Available
- Space ID: A3, Price: \$5.40 | Regular | Status: Available
- Space ID: B1, Price: \$3.60 | Regular | Status: Available




Input

Enter Space ID to Reserve:

Input

×



Enter Space ID to Vacate:

OK

Cancel

Your reservation history:

Space ID: B2, Price: \$7.20 | VIP

Space ID: A1, Price: \$4.50 | Regular

Space ID: A2, Price: \$6.30 | VIP