A

Mini Project On

# **“Parking System”**

Submitted in partially fulfilment for the requirement for the award of the degree of

Bachelor of Technology

in

**ARTIFICIAL INTELLIGENCE &MACHINE LEARNING**

Submitted by

**Deepak Gupta**

**(2202841530015)**

Under the Supervision of

## Prof. Gaurav Narain Singh



Department of Computer Science & Engineering

**UNITED ISTITUTE OF TECHNOLOGY (284)**

(Affiliated to Dr. A.P.J. Abdul Kalam Technical University, Lucknow)

Session -2022-2024

**Vision of the Department**

To be a center of excellence in the field of Computer Science and Engineering for producing talented engineers to ethically serve constantly changing needs of society and industry throughout their career and life.

**Mission of the Department**

**M1.** Accomplish excellence with committed faculty by providing theoretical foundation and practical skills for solving complex engineering problems in the state-of-the-art trends in Computer science and allied disciplines.

**M2.** To foster skills and competency, generating novel ideas, entrepreneurship and model creations focused towards deep knowledge, interpersonal skills and leadership.

**M3.** To develop habitude of research among faculty and students in the area of Computer Science & Allied disciplines by providing the desired environment, for addressing the needs of industry and society.

**M4.** To mould the students with ethical principles in thoughts, expression and deeds.

# Index

**S. No Topic Page No.**

* 1. Introduction

………………………………………………... **4**

* 1. Requirement

……………………………………………….. **5**

* 1. Conceptual Background

…………………………………… **7**

* 1. Implementation

…………………………………………..... **9**

**4.1**  Coding

………………………………………………… **9**

**4.2**  Flow Chart

…………………………………………….. **12**

* 1. User Interface

……………………………………………… **13**

* 1. Future Scope

……………………………………………… **15**

References

…………………………………………………  **17**

## Chapter-1 Introduction

Welcome to Car Parking System using Java! This system allows you to make your task to park the cars easy. Java is a popular programming language that is widely used for developing parking management systems due to its simplicity, versatility, and platform independence.

The Contactless Parking System is a Java-based application designed to streamline the process of managing parking lots efficiently and with minimal physical contact. With the increasing need for contactless services, especially in light of health and safety concerns, the

Contactless Parking System offers a convenient solution for both parking lot operators and users.

This system analyses for an available space in parking area and on the basis of it gives the output to user. If all slots are full then it will simply tell the user to not park car in that particular area.

It increments the available slots if any user removed his/her car from parking area and decrements slots if any user park his/her car. Overall, a parking system in Java provides a flexible and powerful platform for managing parking spaces, improving traffic flow, and enhancing the overall parking experience for drivers. By leveraging the strengths of Java, developers can create robust and scalable parking systems that meet the unique needs of their users.

It presents a menu with four options: park a car, remove a car, view parked cars, and exit the program. It is very simple with coding point of view as it has only one java class in it. We need to import Scanner class and Array List class. Scanner class used to take input through the keyboard from user while Array List is used to make things simple for adding the cars and deleting them. User needs to provide total number of parking slots and then main menu will be displayed for further operations. In park car method, we will check for availability of slots. If slots are available then user will provide car license number and message will be displayed as “Car parked successfully”.

**Chapter-2**

## Requirement

To create a parking system using Java, you need to consider various requirements to ensure that the system functions efficiently and meets the needs of its users. Here's a breakdown of the requirements you should consider:

**Vehicle Entry and Exit:**

The system should allow to vehicles to enter and exit the parking lot smoothly.

**Parking Slot Management:**

The system needs to manage available parking slots, including differentiating between small, medium, and large vehicle slots.

**Parking Reservation:**

Users may need the option to reserve parking slots in advance, if applicable.

**Reporting and Analytics:**

Generate reports on parking slot usage, revenue generation, and occupancy rates this report help user to get information of their car parking

**Performance:**

The system should perform efficiently, handling a large number of transactions and users simultaneously

**Scalability:**

The system should be scalable to accommodate future growth in the number of users and parking spaces.

**Reliability:**

Ensure the system operates reliably under normal and peak load conditions, minimizing downtime and errors.

**Accessibility:**

The system should be accessible to users with disabilities, adhering to accessibility standards and guidelines.

**Parking Reservation:**

Users may need the option to reserve parking slots in advance, this save the time if user if they reserve parking in advance.

## Hardware Requirement

**Computer:**

A computer with java installed is necessary for running the parking system for allotting a slot for parking.

**Internet connection:**

If the project replies on real time than good band width are required.

**Ram and Rom:**

For accessing this program in your pc, user need to meet our requirement such as min 2gb ram and 100gb SSD storage

### Chapter-3 Conceptual Background

Core Java refers to the fundamental components of the Java programming language, providing the basic building blocks for developing robust and platform-independent applications. It encompasses the essential features and libraries that form the foundation of Java development. Here's a brief description of core Java:

**Syntax and Structure:**

Core Java follows a syntax similar to C++ but with simpler and more intuitive features. It includes basic programming constructs such as variables, data types, operators, control structures (if-else, loops), methods, and classes.

**Object-Oriented Programming (OOP):**

Java is primarily an object-oriented language, which means it revolves around objects and classes. Core Java emphasizes principles like encapsulation, inheritance.

**Platform Independence:**

One of the key features of Java is its platform independence. Core Java achieves this through the Java Virtual Machine (JVM), which allows Java bytecode to run on any platform that has a JVM installed, without the need for recompilation.

**Exception Handling:**

Java includes robust exception handling mechanisms that help developers manage errors and unexpected situations gracefully. Core Java allows developers to handle exceptions using trycatch blocks, throw statements, and custom exception classes.

**Memory Management and Garbage Collection:**

Core Java features automatic memory management through garbage collection. The JVM automatically deallocates memory for objects that are no longer in use, reducing the risk of memory leaks and memory-related errors.

**Reporting and Analytics:**

Generate reports on parking slot usage, revenue generation, and occupancy rates this report help user to get information of their car parking .

**Accessibility:**

The system should be accessible to users with disabilities, adhering to accessibility standards and guidelines.

### Chapter-4 Implementation

**4.1 Coding:**

import java.util.ArrayList; import java.util.Scanner; public class ParkingSystem

{

static int totalSlots, availableSlots;

static ArrayList<String> parkedCars = newArrayList<String>(); public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter the total number of parking slots:"); totalSlots = sc.nextInt(); availableSlots = totalSlots; while (true) {

System.out.println("\nWhat would you like to do?");

System.out.println("1. Park a car");

System.out.println("2. Remove a car");

System.out.println("3. View parked cars"); System.out.println("4. Exit"); int choice = sc.nextInt(); switch (choice)

{ case 1:

parkCar(); break;

case 2: removeCar(); break; case 3:

viewParkedCars(); break; case 4:

System.exit(0);

default:

System.out.println("Invalid choice. Please try again.");

}

}

} public static void parkCar() { if (availableSlots == 0) {

System.out.println("Sorry, there are no available parking slots."); return;

}

Scanner sc = new Scanner(System.in);

System.out.println("Enter the license plate number of the car:"); String licensePlate = sc.nextLine(); parkedCars.add(licensePlate); availableSlots--;

System.out.println("Car parked successfully. Avai slots: " + availables);

}

public static void removeCar() { if (availableSlots == totalSlots) {

System.out.println("There are no parked cars."); return;

}

Scanner sc = new Scanner(System.in);

System.out.println("Enter the license plate number of the car to be removed:"); String licensePlate = sc.nextLine();

if (parkedCars.contains(licensePlate)) { parkedCars.remove(licensePlate);

availableSlots++;

System.out.println("Car removed successfully. Available slots: " + availableSlots);

} else

{

System.out.println("The car is not parked here.");

}

}

public static void viewParkedCars() { if (availableSlots == totalSlots)

{

System.out.println("There are no parked cars."); return;

}

System.out.println("Parked cars:"); for (String licensePlate : parkedCars)

{

System.out.println(licensePlate);

}

}

}

**4.2 Flow Chart:**

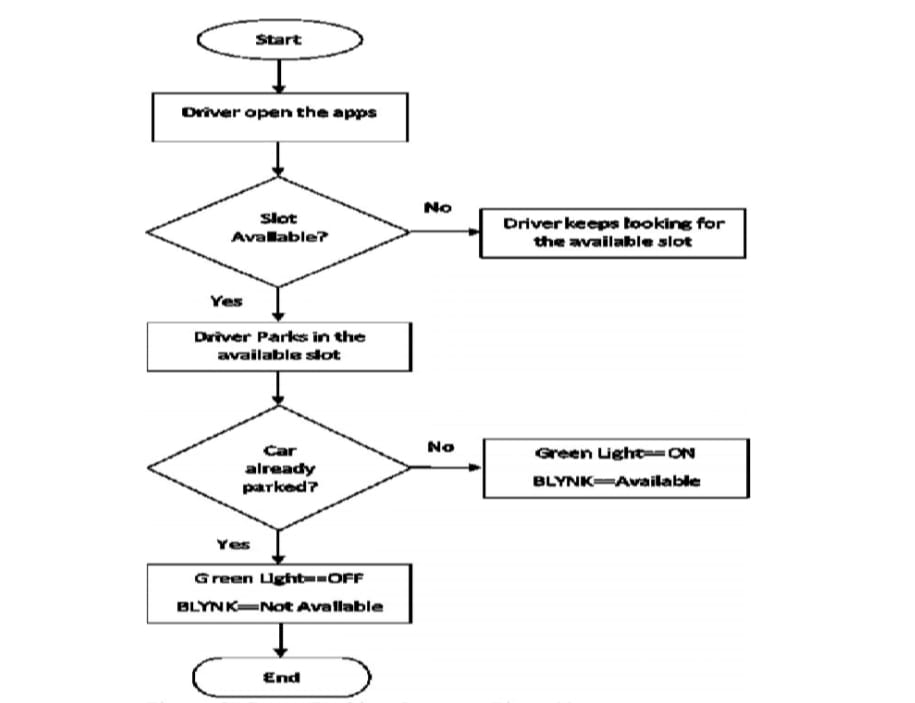
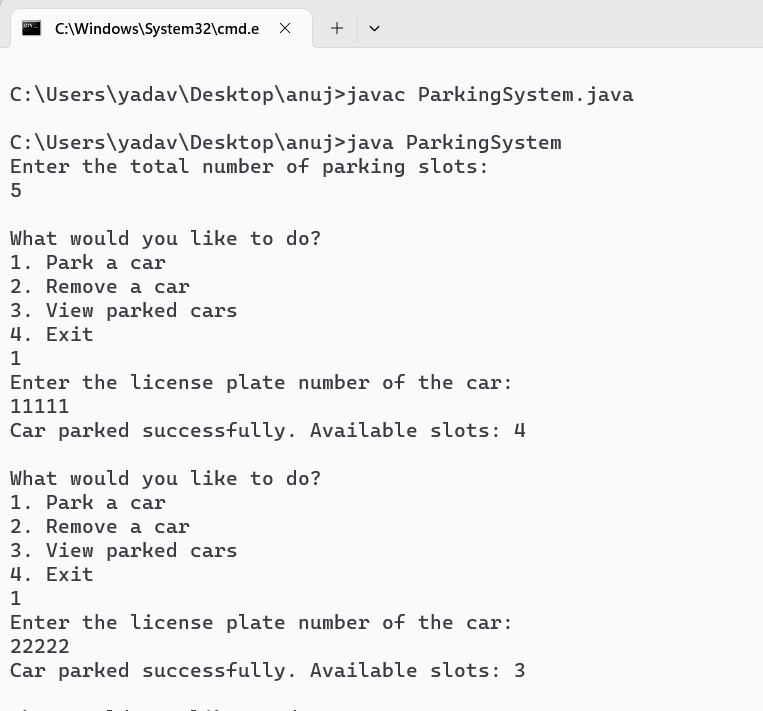
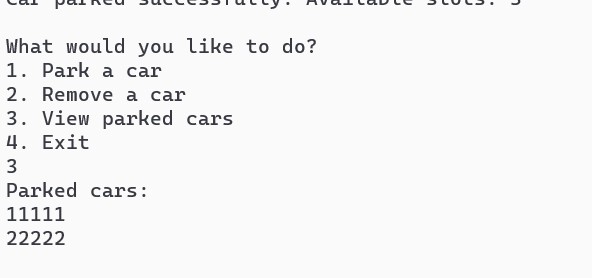


Fig: 4.2 (Sample of flowchart)

### Chapter-5 User Interfaces

 Fig: 5.1 (Sample of user interface)

.



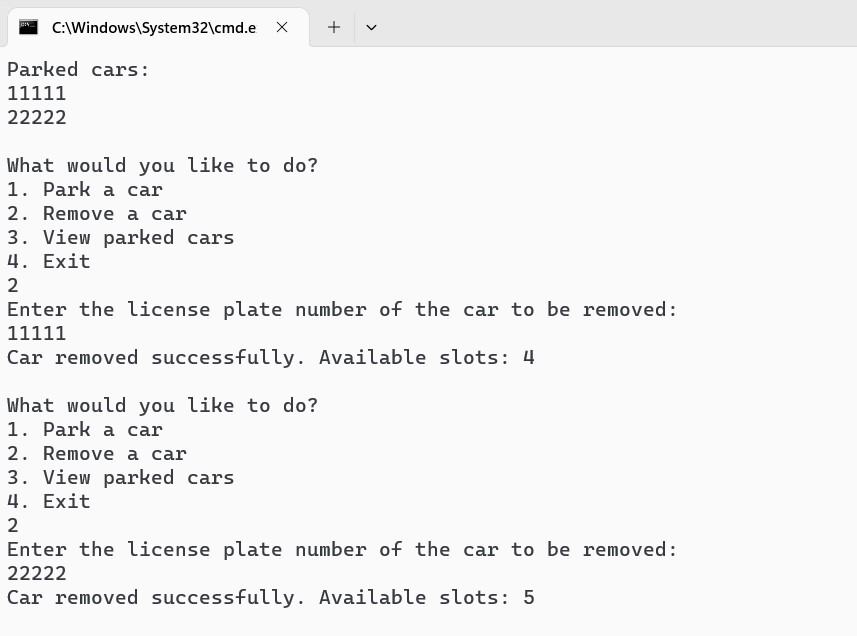


Fig: 5.2 (Sample of user interface)

### Chapter-6 Future Scope

* Implement integration with IoT (Internet of Things) devices such as sensors and cameras to automate parking slot detection, vehicle identification, and surveillance.

* Develop a mobile application for users to find parking spots, reserve parking slots in advance, and make payments using their smartphones.

* Integrate popular payment gateways to facilitate secure online payments for parking fees, including credit/debit cards, mobile wallets, and UPI (Unified Payment Interface).

* Implement smart parking features such as dynamic pricing based on demand, congestion-based routing, and priority parking for electric vehicles or carpooling.

* Design the parking system architecture for scalability to accommodate growing user demands and expanding parking infrastructure.

### References

**Books:**

"Head First Java" by Kathy Sierra and Bert Bates: This book is great for beginners and provides a fun, engaging approach to learning Java.

"Effective Java" by Joshua Bloch: This book is more advanced and provides insights into best practices and advanced features of the Java language.

"Java: A Beginner's Guide" by Herbert Schildt: Another excellent book for beginners, covering the basics of Java programming.

"Java Concurrency in Practice" by Brian Goetz, Tim Peierls, Joshua Bloch, Joseph Bowbeer, David Holmes, and Doug Lea: This book is essential for understanding concurrency in Java applications.

"Java Performance: The Definitive Guide" by Scott Oaks: For those interested in optimizing Java code and understanding performance considerations, this book is invaluable.

Online Resources:

Oracle's Java Tutorials: The official tutorials provided by Oracle cover a wide range of Java topics and are a great resource for beginners and experienced developers alike. Oracle Java Tutorials

Coursera: Coursera offers several Java programming courses taught by instructors from universities and companies like Duke University and Google. Coursera Java Courses

Udemy: Udemy has numerous Java courses catering to different skill levels and topics, from beginner to advanced Java programming and specific frameworks like Spring. Udemy Java Courses

### Appendix

