A MINI PROJECT REPORT
ON
VEHICLE PARKING MANAGEMENT SYSTEM

submitted in partial fulfillment of requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY
in

COMPUTER SCIENCE AND ENGINEERING
by

S.VISHNAVI	(229Y1A05G3)
S.V.RAVI TEJA	(229Y1A05H1)
T.MANASA	(229Y1A05H9)
Y.LOHITHA	(229Y1AO5J4)
P.NEETHUSREE	(239Y5A0514)

under the Esteemed Supervision of
Smt. O.V. Sowmya, MTech., (Ph.D).
Assistant Professor.,
Dept of CSE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

K.S.R.M. COLLEGE OF ENGINEERING

(UGC-Autonomous)
Approved by AICTE , New Delhi and affiliated to JNTUA, Anantapuramu
Accredited by NAAC with A+Grade &B.tech (EEE,ECE,CSE,CE and ME) Programs by NBA
Kadapa, Andhra Pradesh, India– 516 003

2024-2025

K.S.R.M. COLLEGE OF ENGINEERING

(UGC-Autonomous)

Approved by AICTE , New Delhi and affiliated to JNTUA, Anantapuramu

Accredited by NAAC with A+Grade &B.tech (EEE,ECE,CSE,CE and ME) Programs by NBA

Kadapa, Andhra Pradesh, India– 516 003

VISION

To evolve as center of repute for providing quality academic programs amalgamated with creative learning and research excellence to produce graduates with leadership qualities, ethical and human values to serve the nation.

MISSION

M1: To provide high quality education with enriched curriculum blended with impactful teaching-learning practices.

M2: To promote research, entrepreneurship and innovation through industry collaborations.

M3: To produce highly competent professional leaders for contributing to Socio-economic development of region and the nation.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION

To evolve as a recognized center of excellence in the area of Computer Science and Engineering and other related inter-disciplinary fields.

MISSION

M1: To produce competent and industry ready professionals through well balanced curriculum and innovative pedagogy.

M2: To provide conducive environment for research by establishing centre of excellence and industry collaborations.

M3: To install leadership qualities, ethical values among students through various co-curricular and extracurricular activities.

B. TECH. (COMPUTER SCIENCE AND ENGINEERING)

PROGRAM EDUCATIONAL OBJECTIVES

B.Tech - Computer Science and Engineering Program Objectives.

A graduate of the K.S.R.M.C.E, C.S.E should have a successful career in CSE or a related field, and within three to five years, should

PEO1: To excel in their career as competent software engineer in IT and allied organizations.

PEO2: To pursue higher education and to demonstrate research temper for providing solutions to engineering problems.

PEO3: To contribute for the societal development by exhibiting leadership, through professional, social and ethical values.

PROGRAM OUTCOMES

PO1: Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: 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.

PO3: 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.

PO4: 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.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: 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.

PO7: 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.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: 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.

PO11: 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 environment.

PO12: 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 are statements that describe what the graduates of a specific engineering program should be able to do:

PSO1: Professional Skills: The ability to understand, analyze and develop computer programs in the areas related to algorithms, system software, multimedia, web design, big data analytics, and networking for efficient design of computer-based systems of varying complexity.

PSO2: Problem-Solving Skills: The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product for business success.

PSO3: Successful Career and Entrepreneurship: The ability to employ modern computer languages, environments, and platforms in creating innovative career paths to be an entrepreneur, and a zest for higher studies.

COURSE OUTCOMES

CO 1: Design algorithms using appropriate design techniques (divide and conquer, greedy, dynamic programming, etc.,).

CO 2: Implement variety of algorithms such as sorting, searching, graph related, etc., in a high level language.

CO 3: Analyze and compare the performance of algorithms using language features.

CO-PO MAPPING

CourseOutcome	**Program Outcomes**												**Program Specific Outcomes**		
	PO1	PO2	PO3	PO4	PO5	PO6	PO 7	PO8	PO9	PO 10	PO 11	PO 12	PSO1	PSO2	PSO3
CO1	3	2	3		2	1			2	2			3	2	2
CO2	2	2	3		3					2	2		3	3	2
CO3		3		3	3				2		2	2	2	3	2

K.S.R.M. COLLEGE OF ENGINEERING

(UGC-Autonomous)
Approved by AICTE , New Delhi and affiliated to JNTUA, Anantapuramu
Accredited by NAAC with A+Grade &B.tech (EEE,ECE,CSE,CE and ME) Programs by NBA
Kadapa, Andhra Pradesh, India– 516 003

CERTIFICATE

This is to certify that the Design and Analysis of Algorithms Mini project entitled

VEHICLE PARKING MANAGEMENT SYSTEM

is the bonafide work done & submitted by

S.VISHNAVI	**(229Y1A05G3)**
S.V.RAVI TEJA	**(229Y1A05H1)**
T.MANASA	**(229Y1A05H9)**
Y.LOHITHA	**(229Y1A05J4)**
P.NEETHUSREE	**(239Y5A0514)**

in the Department of Computer Science and Engineering,**K.S.R.M.C.E, Kadapa** and is submitted to **Jawaharlal Nehru Technological University Anantapur,** in partial fulfilment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering during 2022-2026.

Supervisor | **Internal Examiner** | **Head of the Department**

Smt. O.V. Sowmya
M.Tech,(Ph.D).
Assistant Professor
Dept of CSE.

A. RamPrakash Reddy
M.Tech,(Ph.D).
Assistant Professor & HOD
Dept of CSE.

DECLARATION

We hereby declare that this Mini Project report titled "**VEHICLE PARKING MANAGEMENT SYSTEM**" is a genuine project work carried out by us, in B. Tech (**Computer Science and Engineering**) degree course of Jawaharlal Nehru Technological University Anantapur and has not been submitted to any other course or University for the award of any degree by us.

Signature of the Student

S. VISHNAVI

S.V. RAVI TEJA

T. MANASA

Y. LOHITHA

P. NEETHUSREE

ACKNOWLEDGEMENTS

An endeavor over a long period can be successful only with the advice and supports of many well-wishers. We take this opportunity to express our gratitude and appreciation to all of them.

We are extremely thankful to our beloved Managing Director **Dr. K. Chandra Obul Reddy** Garu who took keen interest and encouraged us in every effort throughout this course. We are deeply indebted to the supervisor, **Smt. O.V. Sowmya**, M.Tech.,(Ph.D)., Assistant Professor, Department of Computer Science and Engineering for valuable guidance, constructive criticism and keen interest evinced throughout the course of our DAA project work. We are really fortunate to associate ourselves with such an advising and helping guide in every possible way, at all stages, for the successful completion of this work.

We express our deep sense of gratitude to **A. RamPrakash Reddy**, M.Tech., (Ph.D)., Assistant Professor and Head of Department of Computer Science and Engineering for his valuable guidance and constant encouragement given to us during this DAA project and the course.We take this opportunity to express our deep gratitude and appreciation to all those who encouraged us for successfully completion of this Community Service Project work. We wish to express our sincere to gratitude to **Dr.T. Nageswara Prasad**, M.Tech., Ph.D. Vice Principal of K.S.R.M.C.E, Kadapa and **Dr. V.S.S.Murthy**, M.Tech., Ph.D. Principal of K.S.R.M.C.E, Kadapa and for their consistent help and encouragement to complete the DAA project work. We are pleased to express our heart full thanks to our faculty in Department of CSE of KSRMCE for their moral support and good wishes. Finally, we have a notation to express our sincere thanks to friends and all those who guided, inspired and helped us in the completion of our project work.

S. VISHNAVI	**(229Y1A05G3)**
S.V. RAVI TEJA	**(229Y1A05H1)**
T. MANASA	**(229Y1A05H9)**
Y. LOHITHA	**(229Y1A05J4)**
P. NEETHUSREE	**(239Y5A0514)**

INDEX

LIST OF FIGURES

ABSTRACT

The main aim of this project is to reduce the traffic in the parking place. Normally we can see in the multiplexes, cinema halls, large industries, and function halls there is problem they have to go and search which line is empty and which line having place to park the vehicle, for parking then they need workers for parking in correct position it is the money consumed process. So to avoid this problem Parking management System project is implemented.

The Parking Management System is a Python-based application designed to streamline the process of parking vehicle management, including tracking, billing, and space availability in a parking lot. The system supports multiple vehicle types, including cars, bikes, and trucks, and provides functionalities for parking and removing vehicles, checking available spaces, searching and sorting vehicles by various attributes, and generating parking bills. It utilizes an Excel file to log parking transactions, capturing details such as vehicle license plate, owner name, parking time, and cost. The system dynamically adjusts parking charges based on the duration of stay and demand factors (based on parking space occupancy). The program is intended for use in parking facilities where efficient space management and billing are essential, offering an automated solution for both customers and parking lot administrators.

1. INTRODUCTION

The Vehicle Parking Management System is a software application designed to efficiently manage the parking of vehicles in a parking facility. It provides a solution to automate and streamline the process of parking, tracking, and billing for different types of vehicles, including cars, bikes, and trucks. This system is intended for use in parking lots, garages, or any parking facility that needs to manage vehicle occupancy and billing based on usage.

FIG.1.1: Parking Vehicles Slot Wise

The key functionalities of this system include parking and removing vehicles, checking available parking spaces, searching for specific vehicles, and sorting vehicles based on various criteria such as license plate, owner name, or parking duration. Additionally, the system calculates the cost for parking based on the duration of stay and vehicle type, applying a demand-based pricing mechanism. The demand factor adjusts the cost based on the availability of parking spaces — the more crowded the parking lot, the higher the cost.

The system integrates with Microsoft Excel for storing and maintaining parking records, including details such as license plate numbers, vehicle owner information, parking slot assignments, time of parking, duration of stay, and associated charges (base rate, GST, and total cost). The data saved in Excel allows for easy reporting and record-keeping, making it suitable for parking management in both small and medium-sized facilities.

By leveraging Python’s powerful libraries, such as openpyxl for Excel handling and datetime for time-based calculations, this system ensures accurate and reliable management of parking operations, making it a valuable tool for businesses or facilities that require efficient parking management and billing automation.

The Vehicle Parking Management System is designed to automate and manage the parking process for cars, bikes, and trucks in a parking facility. The system addresses key issues such as space allocation, vehicle tracking, and billing, offering an organized solution for parking lot administrators and users. This software is built with Python and integrates with an Excel file to maintain a detailed record of parking transactions.

Key features include:

- Park and remove vehicles based on availability.
- Search for vehicles using license plates or owner names.
- Sort vehicles by various attributes like license plate, owner name, vehicle type, slot, and parking time.
- Demand-based pricing that adjusts parking costs based on space availability.
- Generation of parking bills, including GST calculations, and storage of records in an Excel sheet for easy tracking.

Intelligent Car Parking Management
BLE
US
P
15 free places parking on level 1

FIG.1.2: Parking with Digital Technologies

The Vehicle Parking Management System (VPMS) is a software application developed to efficiently manage parking spaces for different types of vehicles in a parking lot. The system offers features to park vehicles, remove them, check available parking spaces, and view parking history through search and sorting functionalities. It also calculates parking fees based on vehicle type, parking duration, demand-based pricing, and generates reports in an Excel sheet for record-keeping. The system operates in Indian Standard Time (IST) and ensures the proper validation of vehicle number plates and license numbers.

2.MODULES

2.1: System Requirements

Hardware Requirements:

- A computer or laptop with a minimum of 2 GB RAM and 1 GHz processor.
- 200 MB of free disk space to store the software and Excel files.

Software Requirements:

- Python: Version 3.x or higher.
- Libraries:
  - pytz (for timezone handling)
  - openpyxl (for interacting with Excel files)
  - re (for regular expression matching)

2.2: Architecture & Design

System Architecture:

The system follows a **class-based architecture** for better modularity and organization of functionality. The main components of the system are:

1. **ParkingSystem Class**: This is the core class that contains methods for parking vehicles, removing vehicles, displaying available spaces, sorting and searching parked vehicles, and generating reports.
2. **Excel Integration**: The system integrates with the openpyxl library to generate and maintain an Excel file (parking_records.xlsx) for storing parking records such as owner name, vehicle type, number plate, parking time, and cost.
3. **Demand Calculation**: The system calculates demand levels based on the percentage of occupied parking spaces and adjusts the parking fees accordingly.

Components and Features:

1. **Parking Slots**: The system supports different slots for cars, bikes, and trucks, with a fixed capacity for each vehicle type.
2. **Time Zone Management**: The system uses pytz to convert current UTC time to Indian Standard Time (IST) to ensure the parking time is recorded accurately according to IST.
3. **Vehicle Validation**: The system ensures that the number plate and license number of vehicles are valid before parking them.
4. **Fee Calculation**: Parking fees are calculated based on the vehicle type, parking duration, and demand level. GST is added to the total cost.

2.3: Features and Functionalities

1. Parking a Vehicle

- **Function**: park_vehicle(vehicle_type, owner_name, number_plate, license_number)
- **Description**: Parks a vehicle in the parking lot, allocates an available slot, and records the parking time. If the parking lot is full, the system will not allow further parking.
- **Validation**: Ensures that the number plate and license number conform to valid formats. It also checks for duplicate vehicles based on number plates and license numbers.
- **Data Storage**: Parking details are saved in an Excel sheet for further reference.

2. Removing a Vehicle

- **Function**: remove_vehicle(number_plate)
- **Description**: Removes a parked vehicle from the parking lot, calculates the parking fee (including GST), and generates the total cost based on the vehicle type, parking duration, and demand.
- **Data Update**: Updates the Excel sheet with the removal time, cost, GST, and demand factor for the vehicle.

3. Checking Available Spaces

- **Function**: available_spaces()
- **Description**: Displays the number of available parking slots for cars, bikes, and trucks.

4. Displaying Parked Vehicles

- **Function**: display_parked_vehicles()
- **Description**: Displays the list of currently parked vehicles and offers options to search or sort the vehicles.
  - **Sorting**: Vehicles can be sorted by number plate, owner name, or vehicle type.
  - **Search**: The system allows users to search for vehicles by number plate.

5. Demand-Based Pricing

- **Function**: get_demand(vehicle_type)
- **Description**: Calculates the demand level based on the parking occupancy and returns the corresponding pricing multiplier (low, moderate, or high).

6. Generating Reports (Excel Integration)

- **Function**: openpyxl Library
- **Description**: The system generates an Excel file (parking_records.xlsx) that stores parking information such as vehicle owner details, parking time, removal time, slot number, cost, GST, and demand factor.

2.4: Detailed Code Explanation

1. ParkingSystem Class

The ParkingSystem class contains all the methods and properties related to vehicle parking and record management. It has the following key attributes:

- car_space, bike_space, truck_space: The total available parking spaces for cars, bikes, and trucks.
- parking_records: A dictionary to store details of parked vehicles using their number plates as keys.
- number_plate_records, license_number_records: Sets used to track already parked vehicles based on number plates and license numbers.

2 Methods in the ParkingSystem Class

- **is_valid_number_plate()**: Validates the vehicle number plate format using regular expressions.
- **is_valid_license_number()**: Validates the vehicle license number format.
- **find_available_slot()**: Finds the first available slot for a given vehicle type.
- **get_current_time_ist()**: Returns the current time in IST format (converted from UTC).
- **get_demand()**: Calculates demand based on parking space usage and determines the demand multiplier.
- **park_vehicle()**: Handles the parking process, including validation, slot assignment, and record creation.
- **remove_vehicle()**: Handles vehicle removal, fee calculation, and Excel record updating.
- **available_spaces()**: Displays the available parking spaces for each vehicle type.
- **display_parked_vehicles()**: Displays the list of currently parked vehicles with options to sort or search.
- **search_vehicle()**: Allows the user to search for a vehicle by number plate.
- **sort_vehicles()**: Sorts the list of parked vehicles based on a specified criterion.

2.5: Workflow

1 Parking Workflow

1. The user selects the option to park a vehicle.
2. The system validates the vehicle's number plate and license number.
3. It then finds the next available parking slot and parks the vehicle, recording the details in both the system and the Excel file.
4. A confirmation message is shown with the parking details (slot number, parking time, and vehicle details).

2 Vehicle Removal Workflow

1. The user selects the option to remove a vehicle.
2. The system retrieves the vehicle details and calculates the cost based on parking duration and demand.
3. The vehicle is removed from the parking lot, and the records are updated in the Excel file.

3 Checking Available Spaces

1. The user selects the option to check available spaces.
2. The system calculates and displays the number of available parking spaces for cars, bikes, and trucks.

4 Sorting and Searching Parked Vehicles

1. The user selects the option to sort or search parked vehicles.
2. The system provides options to search by vehicle number plate or sort by vehicle type, number plate, or owner name.
3. The sorted or searched vehicle list is displayed to the user.

2.6: User Interface

The system is a command-line interface (CLI) application. The user interacts with the system by entering their choices in a menu-driven format. The available options are:

1. Park Vehicle
2. Remove Vehicle
3. Check Available Spaces
4. Display Parked Vehicles
5. Search Vehicle by Number Plate
6. Sort Vehicles
7. Exit

2.7: Future Enhancements

1. **User Authentication**: Implement a login system to restrict access to certain functionalities.
2. **Mobile App Integration**: Develop a mobile application for better user interaction and notification features.
3. **Dynamic Pricing**: Extend the demand-based pricing model to consider factors like time of day, special events, etc.
4. **Reservation System**: Allow users to reserve parking slots in advance.

3.SOURCE CODE

```
import pytz
from datetime import datetime, timedelta
import re
from openpyxl import Workbook
import time

class ParkingSystem:
    def __init__(self):
        # Initialize parking space configurations
        self.car_space = 50
        self.bike_space = 50
        self.truck_space = 25

        # Initialize records for parked vehicles
        self.parking_records = {}
        self.number_plate_records = set()  # To keep track of number plates
        self.license_number_records = set()  # To keep track of license numbers

        # Initialize parking slots for each vehicle type
        self.car_slots = [None] * 50
        self.bike_slots = [None] * 50
        self.truck_slots = [None] * 25

        # Create an Excel workbook to store parking details
        self.workbook = Workbook()
        self.sheet = self.workbook.active
        self.sheet.append(["Owner Name", "Vehicle Type", "Number Plate", "License
Number", "Parking Time", "Removing Time", "Slot", "Cost", "GST", "Demand Factor",
"Total Cost"])
```

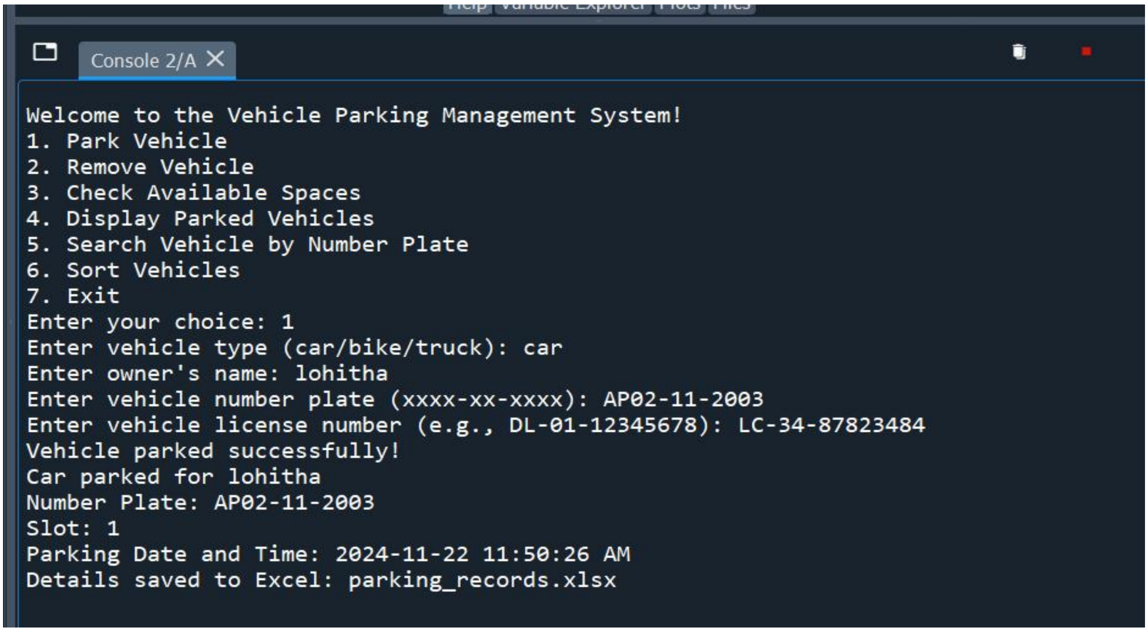
```
    # Define IST timezone using pytz
    self.india_timezone = pytz.timezone('Asia/Kolkata')

def is_valid_number_plate(self, number_plate):
    """Validate vehicle number plate format (xxxx-xx-xxxx)."""
    pattern = r"^[A-Za-z0-9]{4}-[A-Za-z0-9]{2}-[A-Za-z0-9]{4}$"
    return bool(re.match(pattern, number_plate))

def is_valid_license_number(self, license_number):
    """Validate vehicle license number format (XX-XX-XXXXXXXX)."""
    pattern = r"^[A-Za-z]{2}-[0-9]{2}-[0-9]{8}$"
    return bool(re.match(pattern, license_number))

def find_available_slot(self, vehicle_type):
    """Find an available parking slot for a given vehicle type."""
    if vehicle_type == "car":
        for i in range(len(self.car_slots)):
            if self.car_slots[i] is None:
                return i
    elif vehicle_type == "bike":
        for i in range(len(self.bike_slots)):
            if self.bike_slots[i] is None:
                return i
    elif vehicle_type == "truck":
        for i in range(len(self.truck_slots)):
            if self.truck_slots[i] is None:
                return i
    return -1

def get_current_time_ist(self):
    """Get the current IST time in the format 'YYYY-MM-DD hh:mm:ss AM/PM'."""
```

```
        # Get the current time in UTC and convert it to IST using pytz
        utc_time = datetime.now(pytz.utc)  # Get time in UTC
        ist_time = utc_time.astimezone(self.india_timezone)  # Convert UTC time to IST

        # Return time formatted in 12-hour format with AM/PM
        return ist_time.strftime("%Y-%m-%d %I:%M:%S %p")  # %I for 12-hour
format, %p for AM/PM

    def get_demand(self, vehicle_type):
        """Calculate demand level based on parking occupancy and return the
corresponding multiplier."""
        if vehicle_type == "car":
            total_slots = self.car_space
            available_slots = self.car_slots.count(None)
        elif vehicle_type == "bike":
            total_slots = self.bike_space
            available_slots = self.bike_slots.count(None)
        elif vehicle_type == "truck":
            total_slots = self.truck_space
            available_slots = self.truck_slots.count(None)

        used_percentage = ((total_slots - available_slots) / total_slots) * 100

        if used_percentage < 50:
            demand_level = "Low"
            demand_multiplier = 1.0
        elif 50 <= used_percentage <= 75:
            demand_level = "Moderate"
            demand_multiplier = 1.2
        else:
            demand_level = "High"
            demand_multiplier = 1.5
        return demand_multiplier, demand_level

    def park_vehicle(self, vehicle_type, owner_name, number_plate, license_number):
        """Park a vehicle in the parking lot."""
        if not self.is_valid_number_plate(number_plate):
            print("Invalid number plate format. Please use 'xxxx-xx-xxxx' with digits and
alphabets.")
            return

        if not self.is_valid_license_number(license_number):
            print("Invalid license number format. Please use 'XX-XX-XXXXXXXX' (e.g.,
DL-01-12345678).")
            return

        if number_plate in self.number_plate_records:
            print(f"Vehicle with number plate {number_plate} is already parked.")
            return

        if license_number in self.license_number_records:
            print(f"Vehicle with license number {license_number} is already parked.")
            return

        # Find an available slot for the vehicle
        slot = self.find_available_slot(vehicle_type)
        if slot == -1:
            print(f"Sorry, no space available for {vehicle_type}.")
            return

        # Park the vehicle in the slot
        if vehicle_type == "car":
            self.car_slots[slot] = number_plate
        elif vehicle_type == "bike":
            self.bike_slots[slot] = number_plate
        elif vehicle_type == "truck":
            self.truck_slots[slot] = number_plate

        park_time = time.time()
        date_time = self.get_current_time_ist()
        self.parking_records[number_plate] = {
            "owner_name": owner_name,
            "vehicle_type": vehicle_type,
            "slot": slot,
            "park_time": park_time,
            "license_number": license_number,
            "date_time": date_time
        }
        self.number_plate_records.add(number_plate)
        self.license_number_records.add(license_number)

        # Save to Excel on parking
        self.sheet.append([owner_name, vehicle_type, number_plate, license_number,
date_time, None, slot + 1, None, None, None, None])
        self.workbook.save("parking_records.xlsx")

        # Print confirmation
        print(f"Vehicle parked successfully!")
        print(f"{vehicle_type.capitalize()} parked for {owner_name}")
        print(f"Number Plate: {number_plate}")
        print(f"Slot: {slot + 1}")
        print(f"Parking Date and Time: {date_time}")
        print(f"Details saved to Excel: parking_records.xlsx\n")

    def remove_vehicle(self, number_plate):
```

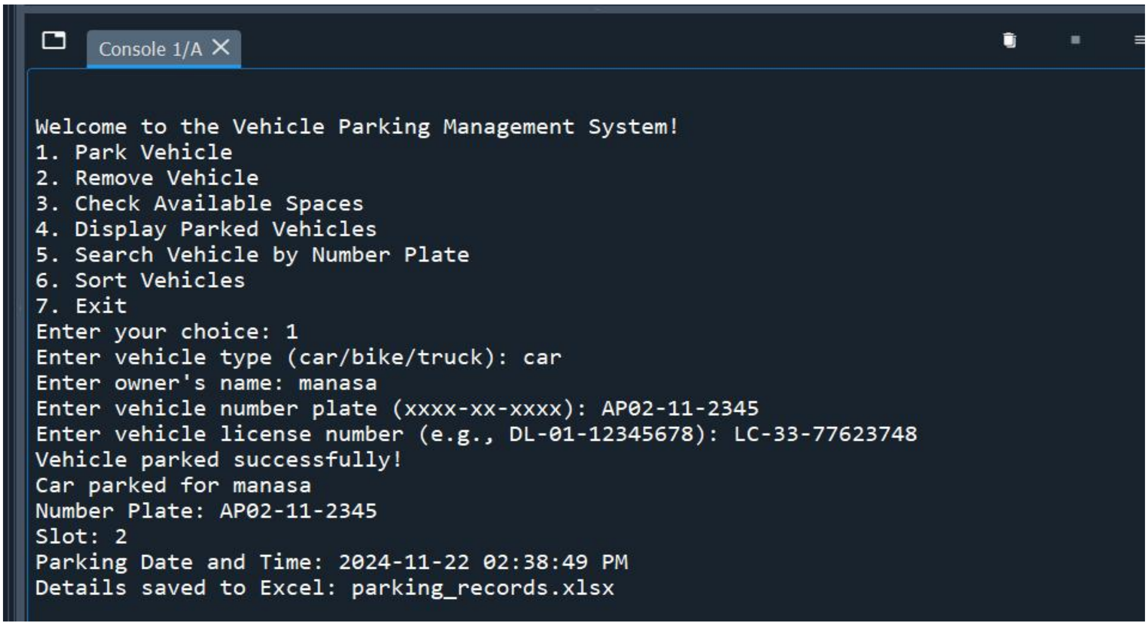
```
"""Remove a vehicle from the parking lot."""
if number_plate not in self.parking_records:
    print(f"No vehicle found with number plate {number_plate}.")
    return

vehicle = self.parking_records.pop(number_plate)
vehicle_type = vehicle["vehicle_type"]
owner_name = vehicle["owner_name"]
slot = vehicle["slot"]
park_time = vehicle["park_time"]
date_time = vehicle["date_time"]

# Calculate demand and cost
demand_multiplier, demand_level = self.get_demand(vehicle_type)
remove_time = time.time()
parked_duration = remove_time - park_time
minutes_parked = round(parked_duration / 60)
hours_parked = minutes_parked // 60
remaining_minutes = minutes_parked % 60

# Pricing based on vehicle type
if vehicle_type == "car":
    base_rate = 50
    self.car_slots[slot] = None
elif vehicle_type == "bike":
    base_rate = 30
    self.bike_slots[slot] = None
elif vehicle_type == "truck":
    base_rate = 100
    self.truck_slots[slot] = None

total_cost = base_rate * (hours_parked + remaining_minutes / 60) *
```

```
demand_multiplier
        gst = total_cost * 0.18
        total_with_gst = total_cost + gst

        # Update Excel with removal details
        remove_time_str = self.get_current_time_ist()
        for row in self.sheet.iter_rows(min_row=2, max_row=self.sheet.max_row):
            if row[2].value == number_plate:
                row[5].value = remove_time_str
                row[7].value = base_rate
                row[8].value = gst
                row[9].value = demand_level
                row[10].value = total_with_gst
                break
        self.workbook.save("parking_records.xlsx")

        print(f"\nVehicle removed successfully!")
        print(f"Owner: {owner_name}")
        print(f"Vehicle Type: {vehicle_type}")
        print(f"Number Plate: {number_plate}")
        print(f"Park Time: {date_time}")
        print(f"Remove Time: {remove_time_str}")
        print(f"Total Cost: {total_with_gst} (Including GST: {gst}, Demand Factor:
{demand_level})")
        print(f"Details updated in Excel: parking_records.xlsx\n")

    def available_spaces(self):
        """Display the available parking spaces."""
        car_available = self.car_space - self.car_slots.count(None)
        bike_available = self.bike_space - self.bike_slots.count(None)
        truck_available = self.truck_space - self.truck_slots.count(None)
        print(f"\nAvailable spaces: ")
        print(f"Car Spaces: {car_available}/{self.car_space}")
        print(f"Bike Spaces: {bike_available}/{self.bike_space}")
        print(f"Truck Spaces: {truck_available}/{self.truck_space}")

    def display_parked_vehicles(self):
        """Display parked vehicles and provide search and sort options."""
        sort_by = input("Enter the field to sort the parked vehicles (number_plate/owner_name/vehicle_type) or press Enter to skip: ").strip().lower()
        if sort_by:
            self.sort_vehicles(sort_by)

        search_query = input("Enter the number plate to search for (leave blank to skip): ").strip().upper()
        if search_query:
            self.search_vehicle(search_query)
        else:
            print("\nCurrently Parked Vehicles:")
            for number_plate, vehicle in self.parking_records.items():
                print(f"Number Plate: {number_plate}, Owner: {vehicle['owner_name']}, Type: {vehicle['vehicle_type']}")

    def search_vehicle(self, number_plate):
        """Search for a vehicle by its number plate."""
        vehicle = self.parking_records.get(number_plate)
        if vehicle:
            print(f"Vehicle found for {vehicle['owner_name']}:")
            print(f"Vehicle Type: {vehicle['vehicle_type']}")
            print(f"License Number: {vehicle['license_number']}")
            print(f"Park Time: {vehicle['date_time']}")
            print(f"Slot: {vehicle['slot'] + 1}")
        else:
            print(f"No vehicle found with number plate {number_plate}.")

    def sort_vehicles(self, sort_by):
        """Sort vehicles based on selected attribute."""
        if sort_by not in ['number_plate', 'owner_name', 'vehicle_type']:
            print("Invalid sort criteria.")
            return

        sorted_vehicles = sorted(self.parking_records.items(), key=lambda x: x[1][sort_by])
        print("Sorted Parked Vehicles:")
        for number_plate, vehicle in sorted_vehicles:
            print(f"Number Plate: {number_plate}, Owner: {vehicle['owner_name']}, Type:
{vehicle['vehicle_type']}")

# Main function
def main():
    parking_system = ParkingSystem()

    while True:
        print("\nWelcome to the Vehicle Parking Management System!")
        print("1. Park Vehicle")
        print("2. Remove Vehicle")
        print("3. Check Available Spaces")
        print("4. Display Parked Vehicles")
        print("5. Search Vehicle by Number Plate")
        print("6. Sort Vehicles")
        print("7. Exit")

        choice = input("Enter your choice: ").strip()

        if choice == "1":
            vehicle_type = input("Enter vehicle type (car/bike/truck): ").strip().lower()
            owner_name = input("Enter owner's name: ").strip()
            number_plate = input("Enter vehicle number plate (xxxx-xx-xxxx):
").strip().upper()
            license_number = input("Enter vehicle license number (e.g., DL-01-12345678):
").strip().upper()
            parking_system.park_vehicle(vehicle_type, owner_name, number_plate,
license_number)
        elif choice == "2":
            number_plate = input("Enter the vehicle number plate to remove:
").strip().upper()
            parking_system.remove_vehicle(number_plate)
        elif choice == "3":
            parking_system.available_spaces()
        elif choice == "4":
            parking_system.display_parked_vehicles()
        elif choice == "5":
            number_plate = input("Enter vehicle number plate to search for: ").strip().upper()
            parking_system.search_vehicle(number_plate)
        elif choice == "6":
            sort_by = input("Enter sort criteria (number_plate/owner_name/vehicle_type):
").strip().lower()
            parking_system.sort_vehicles(sort_by)
        elif choice == "7":
            print("Exiting the system.")
            break
        else:
            print("Invalid choice! Please try again.")

if __name__ == "__main__":
    main()
```

4.INPUT AND OUTPUT SOURCES

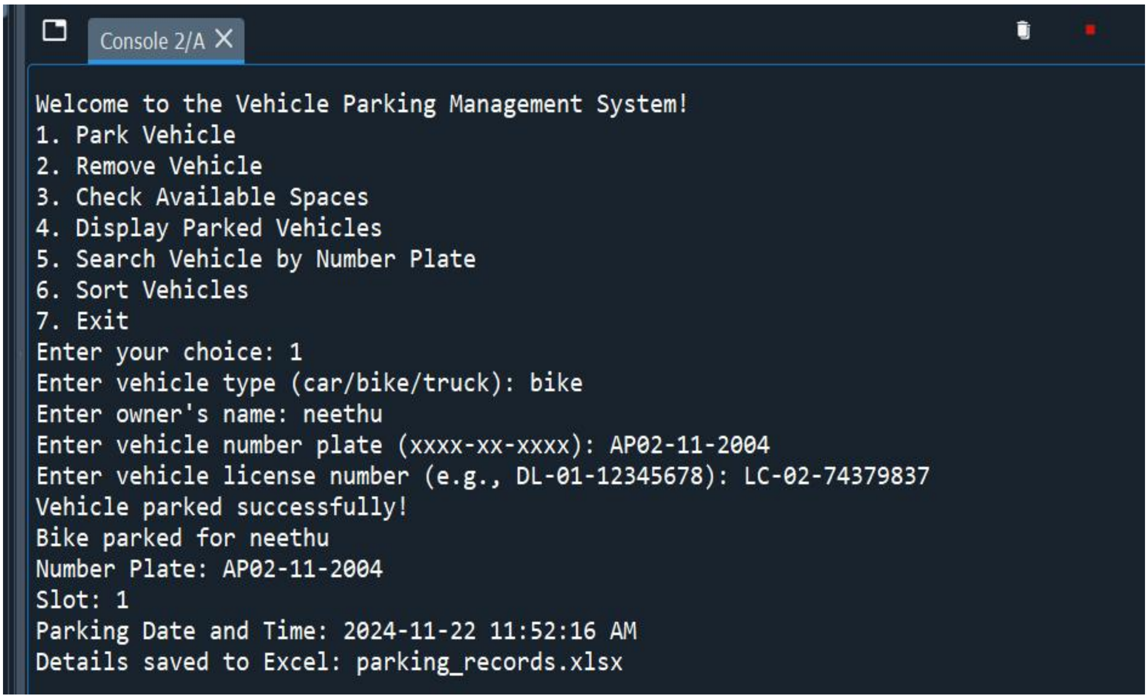


```
Welcome to the Vehicle Parking Management System!
1. Park Vehicle
2. Remove Vehicle
3. Check Available Spaces
4. Display Parked Vehicles
5. Search Vehicle by Number Plate
6. Sort Vehicles
7. Exit
Enter your choice: 1
Enter vehicle type (car/bike/truck): car
Enter owner's name: lohitha
Enter vehicle number plate (xxxx-xx-xxxx): AP02-11-2003
Enter vehicle license number (e.g., DL-01-12345678): LC-34-87823484
Vehicle parked successfully!
Car parked for lohitha
Number Plate: AP02-11-2003
Slot: 1
Parking Date and Time: 2024-11-22 11:50:26 AM
Details saved to Excel: parking_records.xlsx
```

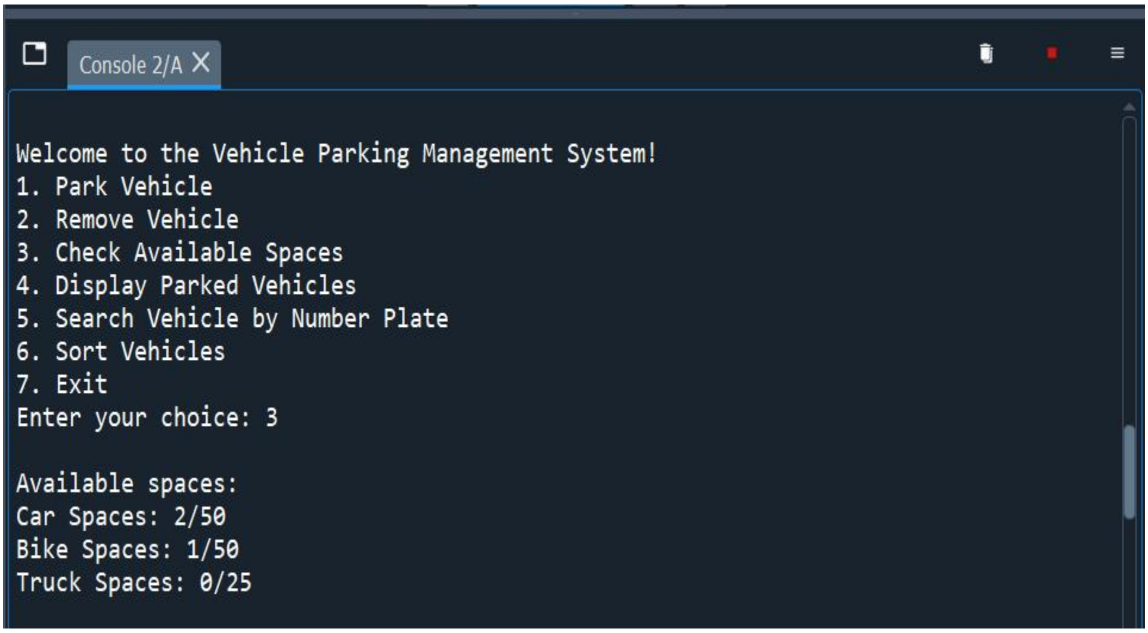


```
Welcome to the Vehicle Parking Management System!
1. Park Vehicle
2. Remove Vehicle
3. Check Available Spaces
4. Display Parked Vehicles
5. Search Vehicle by Number Plate
6. Sort Vehicles
7. Exit
Enter your choice: 1
Enter vehicle type (car/bike/truck): car
Enter owner's name: manasa
Enter vehicle number plate (xxxx-xx-xxxx): AP02-11-2345
Enter vehicle license number (e.g., DL-01-12345678): LC-33-77623748
Vehicle parked successfully!
Car parked for manasa
Number Plate: AP02-11-2345
Slot: 2
Parking Date and Time: 2024-11-22 02:38:49 PM
Details saved to Excel: parking_records.xlsx
```

FIG.4.1: Parking a Different Vehicles

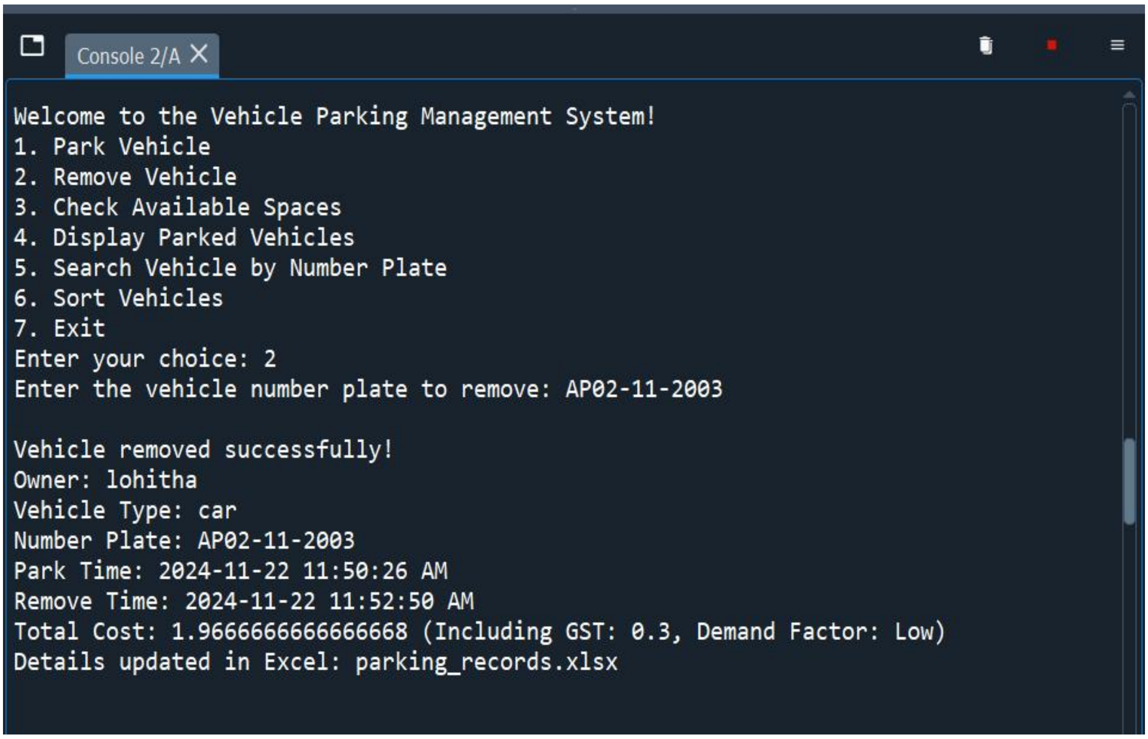


```
Welcome to the Vehicle Parking Management System!
1. Park Vehicle
2. Remove Vehicle
3. Check Available Spaces
4. Display Parked Vehicles
5. Search Vehicle by Number Plate
6. Sort Vehicles
7. Exit
Enter your choice: 1
Enter vehicle type (car/bike/truck): bike
Enter owner's name: neethu
Enter vehicle number plate (xxxx-xx-xxxx): AP02-11-2004
Enter vehicle license number (e.g., DL-01-12345678): LC-02-74379837
Vehicle parked successfully!
Bike parked for neethu
Number Plate: AP02-11-2004
Slot: 1
Parking Date and Time: 2024-11-22 11:52:16 AM
Details saved to Excel: parking_records.xlsx
```



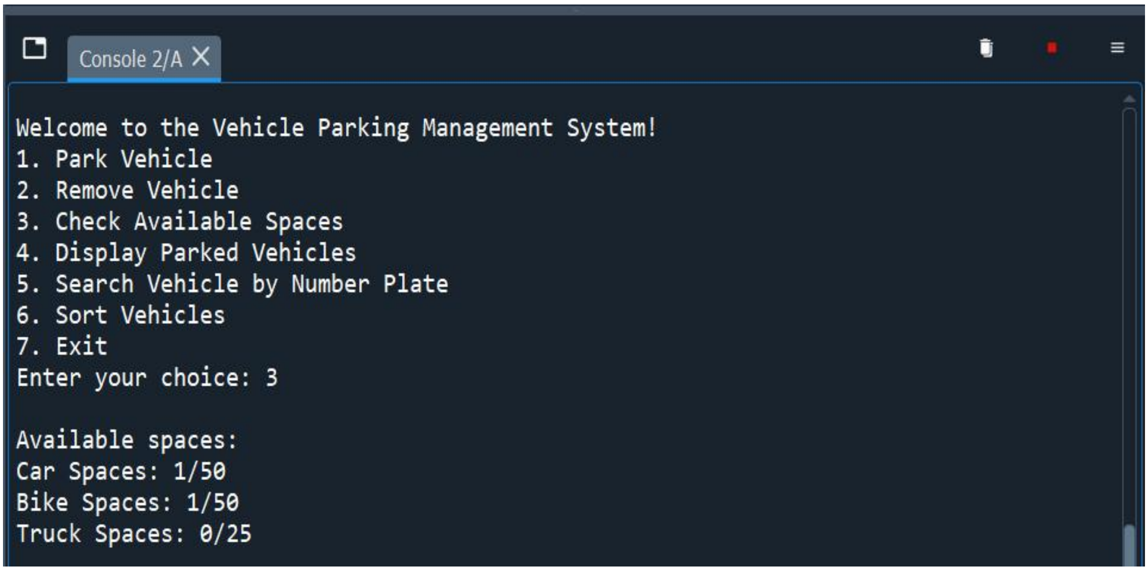
```
Welcome to the Vehicle Parking Management System!
1. Park Vehicle
2. Remove Vehicle
3. Check Available Spaces
4. Display Parked Vehicles
5. Search Vehicle by Number Plate
6. Sort Vehicles
7. Exit
Enter your choice: 3

Available spaces:
Car Spaces: 2/50
Bike Spaces: 1/50
Truck Spaces: 0/25
```

FIG.4.2: Cheking the Available Spaces



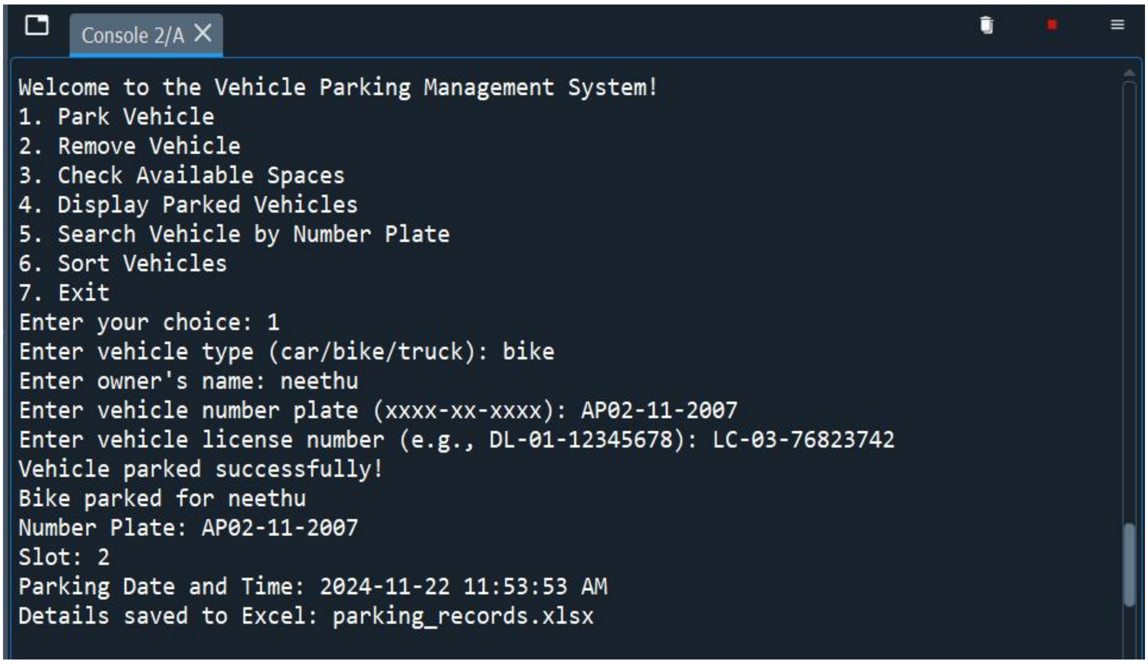
```
Welcome to the Vehicle Parking Management System!
1. Park Vehicle
2. Remove Vehicle
3. Check Available Spaces
4. Display Parked Vehicles
5. Search Vehicle by Number Plate
6. Sort Vehicles
7. Exit
Enter your choice: 2
Enter the vehicle number plate to remove: AP02-11-2003

Vehicle removed successfully!
Owner: lohitha
Vehicle Type: car
Number Plate: AP02-11-2003
Park Time: 2024-11-22 11:50:26 AM
Remove Time: 2024-11-22 11:52:50 AM
Total Cost: 1.9666666666666668 (Including GST: 0.3, Demand Factor: Low)
Details updated in Excel: parking_records.xlsx
```

FIG.4.3: Removing the Vehicle



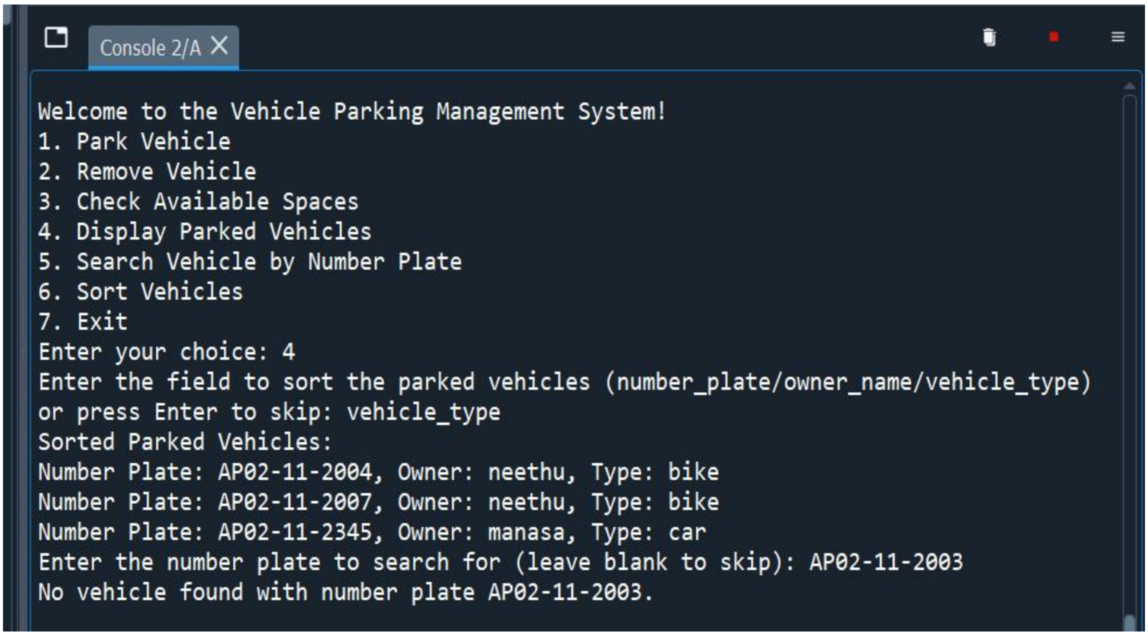
```
Welcome to the Vehicle Parking Management System!
1. Park Vehicle
2. Remove Vehicle
3. Check Available Spaces
4. Display Parked Vehicles
5. Search Vehicle by Number Plate
6. Sort Vehicles
7. Exit
Enter your choice: 3

Available spaces:
Car Spaces: 1/50
Bike Spaces: 1/50
Truck Spaces: 0/25
```

FIG.4.4: Checking the Available Spaces



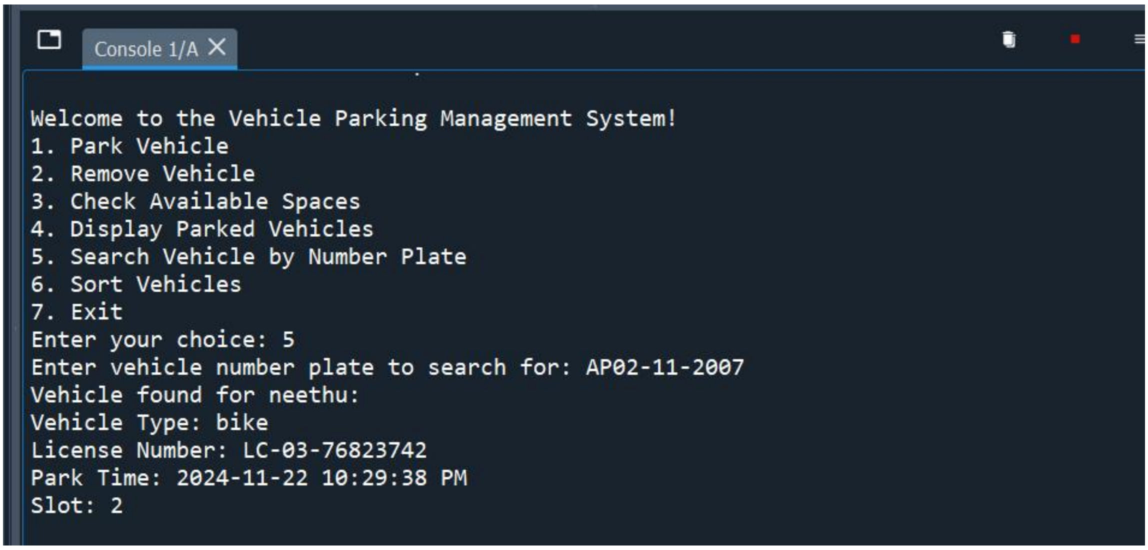
```
Welcome to the Vehicle Parking Management System!
1. Park Vehicle
2. Remove Vehicle
3. Check Available Spaces
4. Display Parked Vehicles
5. Search Vehicle by Number Plate
6. Sort Vehicles
7. Exit
Enter your choice: 1
Enter vehicle type (car/bike/truck): bike
Enter owner's name: neethu
Enter vehicle number plate (xxxx-xx-xxxx): AP02-11-2007
Enter vehicle license number (e.g., DL-01-12345678): LC-03-76823742
Vehicle parked successfully!
Bike parked for neethu
Number Plate: AP02-11-2007
Slot: 2
Parking Date and Time: 2024-11-22 11:53:53 AM
Details saved to Excel: parking_records.xlsx
```

FIG.4.5: Parking a Different Vehicle Type



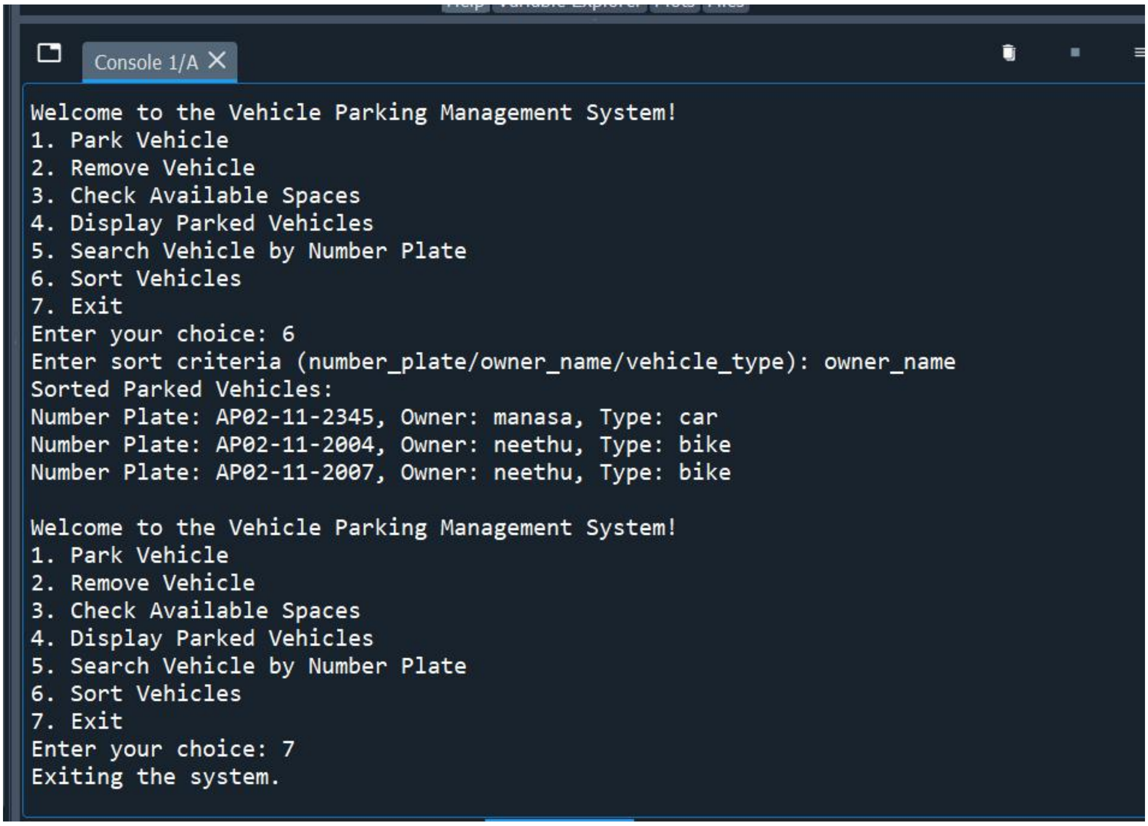
```
Welcome to the Vehicle Parking Management System!
1. Park Vehicle
2. Remove Vehicle
3. Check Available Spaces
4. Display Parked Vehicles
5. Search Vehicle by Number Plate
6. Sort Vehicles
7. Exit
Enter your choice: 4
Enter the field to sort the parked vehicles (number_plate/owner_name/vehicle_type)
or press Enter to skip: vehicle_type
Sorted Parked Vehicles:
Number Plate: AP02-11-2004, Owner: neethu, Type: bike
Number Plate: AP02-11-2007, Owner: neethu, Type: bike
Number Plate: AP02-11-2345, Owner: manasa, Type: car
Enter the number plate to search for (leave blank to skip): AP02-11-2003
No vehicle found with number plate AP02-11-2003.
```

FIG.4.6: Displaying the Parked Vehicles



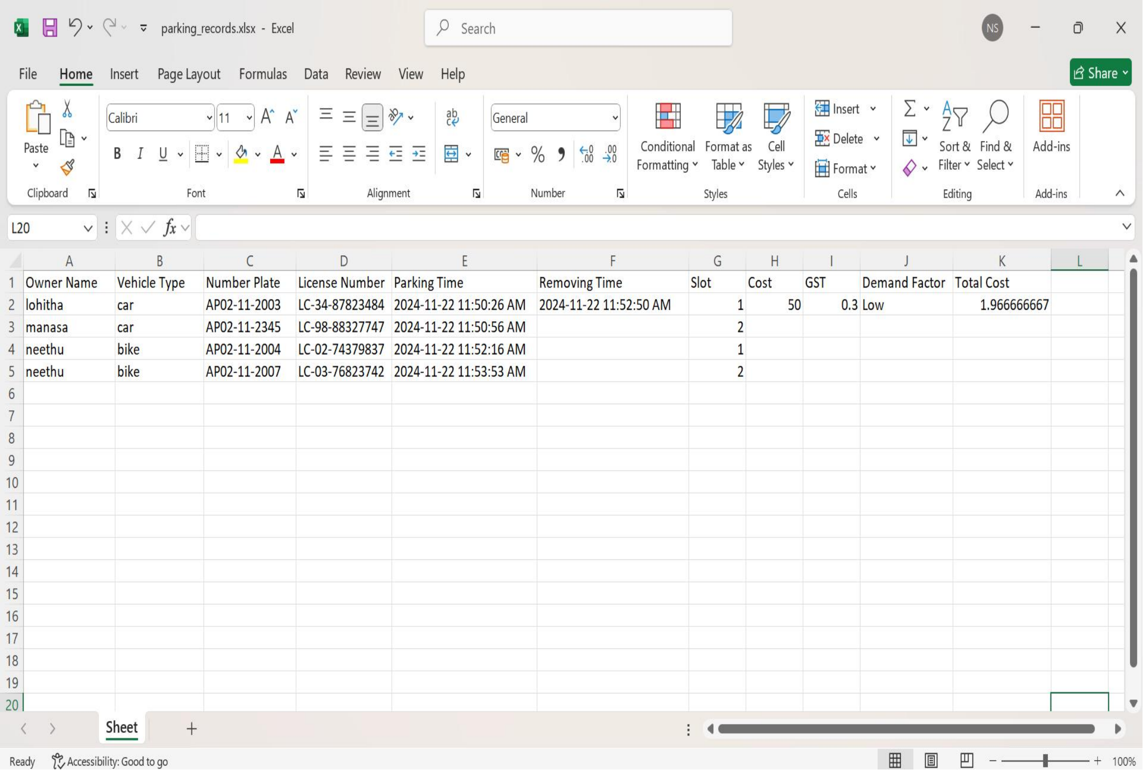
```
Welcome to the Vehicle Parking Management System!
1. Park Vehicle
2. Remove Vehicle
3. Check Available Spaces
4. Display Parked Vehicles
5. Search Vehicle by Number Plate
6. Sort Vehicles
7. Exit
Enter your choice: 5
Enter vehicle number plate to search for: AP02-11-2007
Vehicle found for neethu:
Vehicle Type: bike
License Number: LC-03-76823742
Park Time: 2024-11-22 10:29:38 PM
Slot: 2
```

FIG.4.7: Searching the Vehicles



```
Welcome to the Vehicle Parking Management System!
1. Park Vehicle
2. Remove Vehicle
3. Check Available Spaces
4. Display Parked Vehicles
5. Search Vehicle by Number Plate
6. Sort Vehicles
7. Exit
Enter your choice: 6
Enter sort criteria (number_plate/owner_name/vehicle_type): owner_name
Sorted Parked Vehicles:
Number Plate: AP02-11-2345, Owner: manasa, Type: car
Number Plate: AP02-11-2004, Owner: neethu, Type: bike
Number Plate: AP02-11-2007, Owner: neethu, Type: bike

Welcome to the Vehicle Parking Management System!
1. Park Vehicle
2. Remove Vehicle
3. Check Available Spaces
4. Display Parked Vehicles
5. Search Vehicle by Number Plate
6. Sort Vehicles
7. Exit
Enter your choice: 7
Exiting the system.
```

FIG.4.8: Sorting the Vehicles



Owner Name	Vehicle Type	Number Plate	License Number	Parking Time	Removing Time	Slot	Cost	GST	Demand Factor	Total Cost
lohitha	car	AP02-11-2003	LC-34-87823484	2024-11-22 11:50:26 AM	2024-11-22 11:52:50 AM	1	50	0.3	Low	1.966666667
manasa	car	AP02-11-2345	LC-98-88327747	2024-11-22 11:50:56 AM		2				
neethu	bike	AP02-11-2004	LC-02-74379837	2024-11-22 11:52:16 AM		1				
neethu	bike	AP02-11-2007	LC-03-76823742	2024-11-22 11:53:53 AM		2				

FIG.4.9: Saving the database into Excel Sheet

5.CONCLUSION

The Vehicle Parking Management System you’ve developed is a robust and functional solution for managing parking spaces and vehicle records efficiently. It handles multiple vehicle types (cars, bikes, trucks) and ensures proper validation, availability checks, and accurate tracking of parking details. The system includes several important features, such as:

- **Parking Management:** Vehicles can be parked and removed with ease. The system ensures that only valid vehicles are allowed to park, preventing duplicate entries for the same number plate or license number.
- **Dynamic Slot Allocation:** The parking slots are dynamically assigned based on the availability of spaces for each vehicle type (car, bike, truck). This ensures optimal space utilization.
- **Cost and Demand Calculations:** The system calculates parking costs based on vehicle type, parking duration, and demand (affected by occupancy levels). This is a practical approach to generating fair parking fees.
- **Excel Integration:** The system leverages Excel for maintaining detailed records, making it easy to track past parking transactions and analyze vehicle usage trends.
- **User-Friendly Interface:** The program provides an interactive console interface that allows users to park vehicles, remove them, check available spaces, search for parked vehicles, and sort them based on various criteria.
- **Time and Date Management:** The system utilizes IST (Indian Standard Time) for accurate time tracking, ensuring that all transactions (parking and removal) are logged with the correct time zone.
- **Pricing for Short Durations**: The cost calculation could be refined to avoid charging disproportionately for very short parking durations, rounding durations to the nearest minute or even offering a minimum charge.

With features like demand-based pricing, GST calculation, and real-time slot tracking, this system is tailored to address the challenges faced in modern parking management. The use of Excel for data storage ensures that records are preserved for long-term analysis and reporting.

This implementation emphasizes accuracy, modularity, and scalability, making it suitable for a variety of scenarios, from small parking facilities to large commercial spaces. By automating manual processes, the system saves time, reduces errors, and enhances user satisfaction.

6.REFERENCES

[1]. W3Schools Python Tutorial - https://www.w3schools.com/python/

[2]. GeeksforGeeks - https://www.geeksforgeeks.org

[3]. Real Python - https://realpython.com

[4]. Stack Overflow - https://stackoverflow.com

[5]. GitHub Repositories - https://github.com

[6]. Regex Library (Python re) - https://docs.python.org/3/library/re.html

[7]. Datetime Module (Python) - https://docs.python.org/3/library/datetime.html

[8]. OpenPyXL Documentation - https://openpyxl.readthedocs.io

[9]. pytz Documentation - https://pytz.sourceforge.net

[10]. Python Documentation - https://docs.python.org