1.PROBLEM STATEMENT:

Design and implement a Parking Lot Management System using Python that efficiently manages the parking slots,vehicle information and payment processing.

2.INTRODUCTION:

In an era where urbanization and vehicle ownership are on the rise, effective parking management has become increasingly vital. The Parking Lot Management System addresses the challenges of managing parking spaces in a systematic and efficient manner.

This application is tailored for parking lot administrators, enabling them to oversee vehicle entry and exit seamlessly. By providing a clear and organized interface, the system helps minimize congestion and confusion often associated with parking lots, ensuring a smoother experience for both operators and drivers.

This project aims to develop a Python-based parking lot management system. The system will allow users to register their vehicles, view vehicle information, and remove their vehicles from the lot. Upon registration, the system will validate the vehicle type and license plate number, ensuring no duplicates are allowed. Vehicle information, including type, license plate, and entry time, will be accessible to users. When removing a vehicle, the system will calculate the parking fee based on the duration of stay and update the database accordingly. The system will provide a user-friendly interface and error handling mechanisms for a seamless user experience.

This project simulates a simple parking lot management system. It allows users to:

- Add vehicles of different types (car, truck, motorcycle) to the parking lot.
- Remove vehicles and calculate parking fares based on the duration of their stay.
- View information about vehicles parked in specific spots (by row and column) or by their plate number.
- Display a visual representation of the parking lot, indicating occupied and empty spots.

The code utilizes Python classes to represent the parking lot, vehicles, and their associated information. The program offers a user-friendly menu interface for interacting with the parking lot functionalities.

This project demonstrates fundamental programming concepts like:

- Object-Oriented Programming (OOP) with classes and methods
- Data structures (lists, dictionaries) for efficient data storage and retrieval

- User input and output management
- Basic calculations for parking fare

This is a basic implementation and can be further extended to include features like:

- Different parking rate structures (flat rate, hourly rates)
- Time validation for vehicle entry and exit times
- Reservation system for parking spots
- Integration with payment processing systems

Key Features:

1. **User-Friendly Interface**: The system offers an intuitive menu-driven interface, making it easy for users to navigate through various functions, whether it's adding or removing vehicles or viewing parking availability.

2. **Real-Time Availability Tracking**: Users can instantly see available spots, enhancing the parking experience and reducing the time spent searching for parking.

3. **Comprehensive Vehicle Management**: Administrators can add vehicles with details like type and license plate number, view information about parked vehicles, and efficiently remove them upon exit.

4. **Automated Fare Calculation**: The system calculates parking fees based on the duration of stay, providing a clear breakdown of costs that enhances transparency and trust.

5. **Detailed Reporting**: Administrators can access summaries of parked vehicles and collected fares, allowing for better financial oversight and planning.

6. **Input Validation**: The application includes robust input validation to prevent errors, ensuring that users provide valid information when interacting with the system.

7. **Future Expansion Potential**: The modular design of the system allows for future enhancements, such as dynamic rate adjustments, vehicle search features, and data persistence options, making it adaptable to evolving needs.

CODE BREAKDOWN:

1. Class Definition: ParkingLot

The ParkingLot class encapsulates all functionalities related to managing the parking lot.

- **Attributes:**
    - num_rows: Number of rows in the parking lot.
    - num_cols: Number of columns in the parking lot.
    - parking_spots: A 2D list representing parking spots, initialized as empty spaces (' ').
    - available_spots: A counter for available parking spots.
    - vehicle_info: A dictionary storing details about parked vehicles (plate number, type, entry time, position).

2. Initialization Method: __init__

This method initializes the parking lot with the specified number of rows and columns, creating an empty structure for parking spots.

```
def __init__(self, num_rows, num_cols):
    self.num_rows = num_rows
    self.num_cols = num_cols
    self.parking_spots = [[' ' for _ in range(num_cols)] for _ in range(num_rows)]
    self.available_spots = num_rows * num_cols
    self.vehicle_info = {}
```

3. Display Method: display_parking_lot

This method prints a visual representation of the parking lot, showing occupied spots with vehicle types and plate numbers, and empty spots clearly.

```
def display_parking_lot(self):
    print("|--------------------|")
    for row in range(self.num_rows):
        print("|", end="")
        for col in range(self.num_cols):
            spot = self.parking_spots[row][col]

 if spot != ' ':
```

```
            vehicle = self.vehicle_info[spot]
            display_text = f"{vehicle['type'][0]}({spot})"
            print(f"[{display_text: <10}]", end='')
        else:
            print("[          ]", end='')
    print("|")
print("|--------------------|")
```

4. Adding a Vehicle: add_vehicle

This method allows the addition of a vehicle to a specific spot, provided that the spot is empty and the plate number is unique. It records the entry time and updates available spots.

```
def add_vehicle(self, vehicle_type, plate_number, row, col):
    if self.parking_spots[row][col] == ' ' and plate_number not in self.vehicle_info:
        self.parking_spots[row][col] = plate_number
        self.available_spots -= 1
        entry_time = datetime.datetime.now()
        self.vehicle_info[plate_number] = {"type": vehicle_type, "entry_time": entry_time, "position": (row, col)}
        print("Vehicle Added to Lot!")
        print("Time Entered:", entry_time.strftime("%H:%M:%S"))
        print("SPOTS AVAILABLE:", self.available_spots)
        time.sleep(2)
    else:
        ...
```

5. Removing a Vehicle: remove_vehicle

This method removes a vehicle from a specified spot, calculates the parking fee based on the time parked, and updates the available spots.

```
def remove_vehicle(self, row, col):
    if self.parking_spots[row][col] != ' ':
        plate_number = self.parking_spots[row][col]
        if plate_number in self.vehicle_info:
```

```
        exit_time = datetime.datetime.now()
        entry_time = self.vehicle_info[plate_number]["entry_time"]
        elapsed_time = exit_time - entry_time
        hours_parked = elapsed_time.total_seconds() / 3600
        fare = calculate_fare(hours_parked)
        ...
```

6. Viewing Vehicle Information: view_vehicle_info and view_vehicle_by_plate

These methods allow users to retrieve details about a vehicle based on its parking spot or plate number.

```
def view_vehicle_info(self, row, col):
    if self.parking_spots[row][col] != ' ':
        plate_number = self.parking_spots[row][col]
        if plate_number in self.vehicle_info:
            ...
```

7. Fare Calculation Function: calculate_fare

This standalone function computes the parking fee based on the hours parked, applying a base rate and an hourly rate.

```
def calculate_fare(hours_parked):
    base_rate = 5.0
    hourly_rate = 2.0
    fare = base_rate + (hours_parked * hourly_rate)
    return round(fare, 2)
```

8. Main Function: main

The main() function orchestrates user interaction, offering a menu to execute different functionalities like adding/removing vehicles, viewing information, and displaying the parking lot.

```
def main():
    num_rows = 4
    num_cols = 5
    parking_lot = ParkingLot(num_rows, num_cols)
```

```
    while True:
        ...
        if choice == 1:
            ...
        elif choice == 2:
            ...
        ...
```

3.IMPLEMENTATION:

```
import datetime
import time

class ParkingLot:
    def __init__(self, num_rows, num_cols):
        self.num_rows = num_rows
        self.num_cols = num_cols
        self.parking_spots = [[' ' for _ in range(num_cols)] for _ in range(num_rows)]
        self.available_spots = num_rows * num_cols
        self.vehicle_info = {}

    def display_parking_lot(self):
        print("|--------------------|")
        for row in range(self.num_rows):
            print("|", end="")
            for col in range(self.num_cols):
                spot = self.parking_spots[row][col]
                if spot != ' ':
                    # Display vehicle type and plate number
```

```
            vehicle = self.vehicle_info[spot]
                display_text = f"{vehicle['type'][0]}({spot})"  # Display type initial and plate number
                print(f"[{display_text: <10}]", end='')  # Align to 10 characters
            else:
                print("[          ]", end='')  # Empty spot
        print("|")
    print("|--------------------|")

def add_vehicle(self, vehicle_type, plate_number, row, col):
    if self.parking_spots[row][col] == ' ' and plate_number not in self.vehicle_info:
        self.parking_spots[row][col] = plate_number  # Store plate number directly
        self.available_spots -= 1
        entry_time = datetime.datetime.now()
        self.vehicle_info[plate_number] = {"type": vehicle_type, "entry_time": entry_time, "position": (row, col)}
        print("Vehicle Added to Lot!")
        print("Time Entered:", entry_time.strftime("%H:%M:%S"))
        print("SPOTS AVAILABLE:", self.available_spots)
        time.sleep(2)  # Wait for 2 seconds
    else:
        if plate_number in self.vehicle_info:
            print("Duplicate plate number. Vehicle cannot be added.")
        else:
            print("Spot is already occupied.")

def remove_vehicle(self, row, col):
    if self.parking_spots[row][col] != ' ':
        plate_number = self.parking_spots[row][col]  # Now we store the plate number directly in the spot
```

```
        if plate_number in self.vehicle_info:
            exit_time = datetime.datetime.now()
            entry_time = self.vehicle_info[plate_number]["entry_time"]
            elapsed_time = exit_time - entry_time
            hours_parked = elapsed_time.total_seconds() / 3600
            fare = calculate_fare(hours_parked)
            print("Vehicle Removed from Lot!")
            print("Time Exited:", exit_time.strftime("%H:%M:%S"))
            print("Fare: $", fare)
            print("SPOTS AVAILABLE:", self.available_spots + 1)
            del self.vehicle_info[plate_number]
            self.parking_spots[row][col] = ' '  # Clear the spot
            self.available_spots += 1
        else:
            print("No vehicle parked in that spot.")
    else:
        print("No vehicle parked in that spot.")


def view_vehicle_info(self, row, col):
    if self.parking_spots[row][col] != ' ':
        plate_number = self.parking_spots[row][col]  # Use plate number directly
        if plate_number in self.vehicle_info:
            vehicle_type = self.vehicle_info[plate_number]["type"]
            entry_time = self.vehicle_info[plate_number]["entry_time"]
            print("Vehicle Type:", vehicle_type)
            print("Plate Number:", plate_number)
            print("Entry Time:", entry_time.strftime("%H:%M:%S"))
        else:
            print("No vehicle parked in that spot.")
```

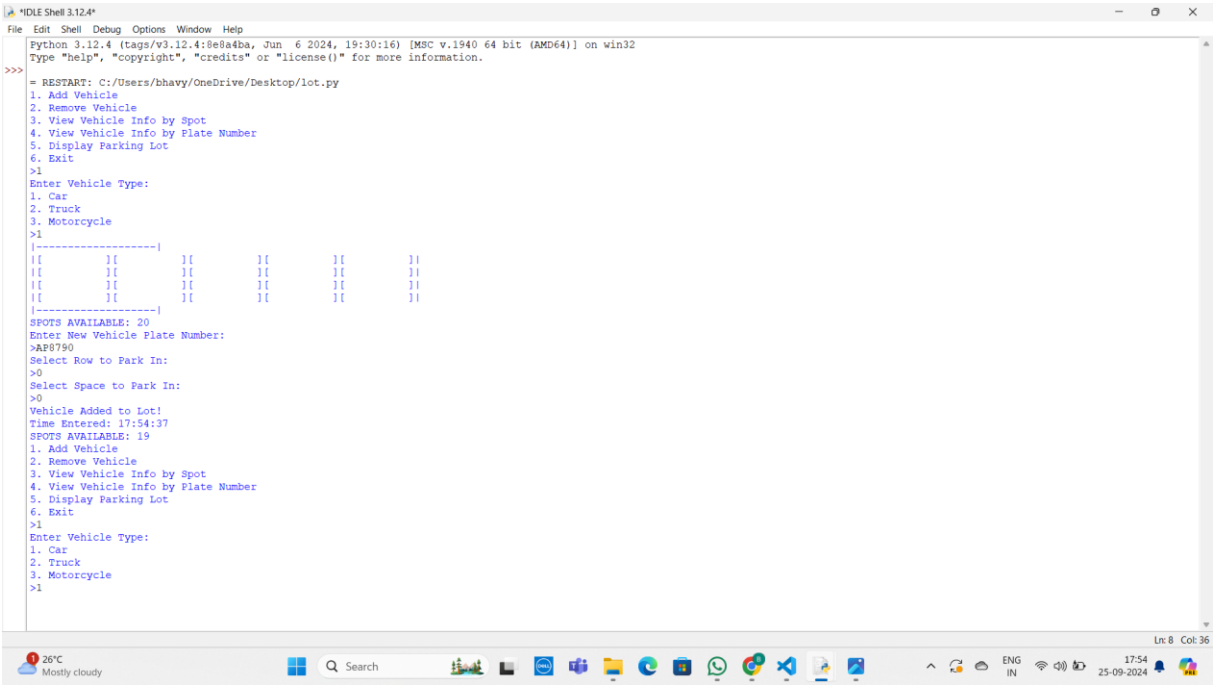
```
        else:
            print("No vehicle parked in that spot.")

    def view_vehicle_by_plate(self, plate_number):
        if plate_number in self.vehicle_info:
            vehicle_type = self.vehicle_info[plate_number]["type"]
            entry_time = self.vehicle_info[plate_number]["entry_time"]
            row, col = self.vehicle_info[plate_number]["position"]
            print("Vehicle Found!")
            print("Vehicle Type:", vehicle_type)
            print("Plate Number:", plate_number)
            print("Entry Time:", entry_time.strftime("%H:%M:%S"))
            print(f"Parked at Row: {row}, Column: {col}")
        else:
            print(f"No vehicle with plate number '{plate_number}' found.")

def calculate_fare(hours_parked):
    base_rate = 5.0
    hourly_rate = 2.0
    fare = base_rate + (hours_parked * hourly_rate)
    return round(fare, 2)

def main():
    num_rows = 4
    num_cols = 5
    parking_lot = ParkingLot(num_rows, num_cols)

    while True:
        print("1. Add Vehicle")
```

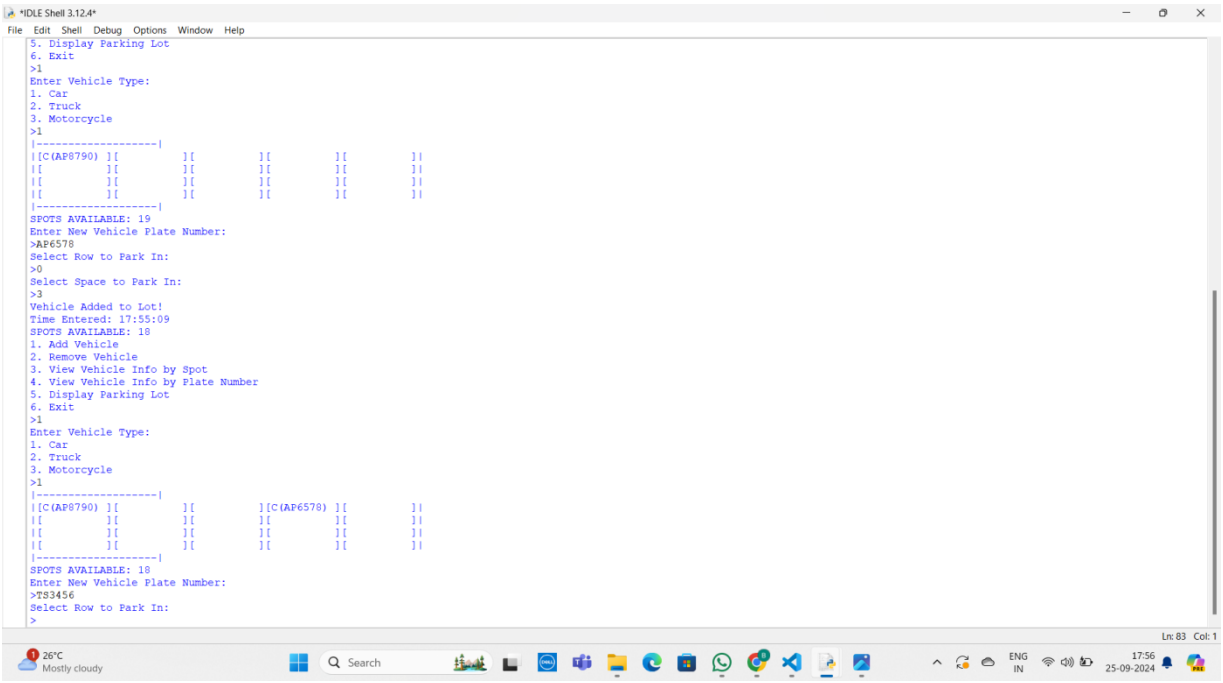
```
print("2. Remove Vehicle")
print("3. View Vehicle Info by Spot")
print("4. View Vehicle Info by Plate Number")
print("5. Display Parking Lot")
print("6. Exit")
choice = int(input(">"))

if choice == 1:
    print("Enter Vehicle Type:")
    print("1. Car")
    print("2. Truck")
    print("3. Motorcycle")
    vehicle_type_input = int(input(">"))
    vehicle_types = {1: "Car", 2: "Truck", 3: "Motorcycle"}
    vehicle_type = vehicle_types.get(vehicle_type_input, None)

    if vehicle_type is None:
        print("Invalid vehicle type.")
        continue

    parking_lot.display_parking_lot()
    print("SPOTS AVAILABLE:", parking_lot.available_spots)

    plate_number = input("Enter New Vehicle Plate Number:\n>")

    row = int(input("Select Row to Park In:\n>"))
    col = int(input("Select Space to Park In:\n>"))

    parking_lot.add_vehicle(vehicle_type, plate_number, row, col)
```

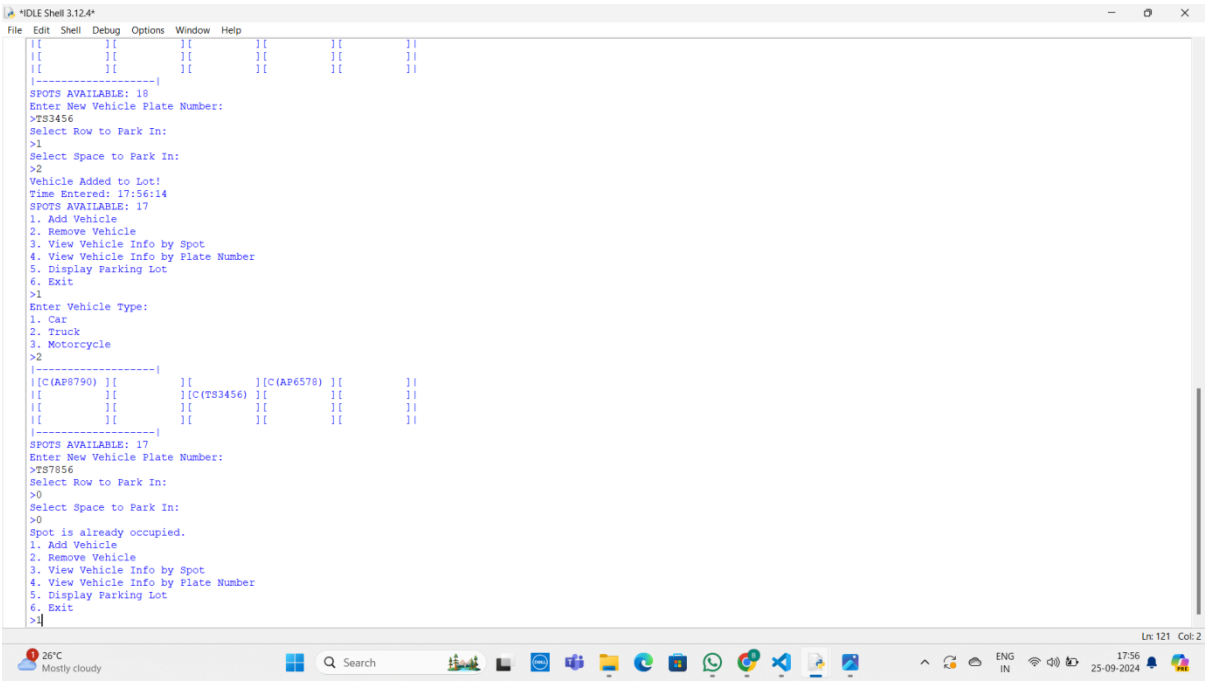
```
elif choice == 2:
    parking_lot.display_parking_lot()
    print("SPOTS AVAILABLE:", parking_lot.available_spots)

    row = int(input("Select Row to Remove Vehicle From:\n>"))
    col = int(input("Select Space to Remove Vehicle From:\n>"))

    parking_lot.remove_vehicle(row, col)

elif choice == 3:
    parking_lot.display_parking_lot()
    print("SPOTS AVAILABLE:", parking_lot.available_spots)

    row = int(input("Select Row to View Vehicle Info:\n>"))
    col = int(input("Select Space to View Vehicle Info:\n>"))

    parking_lot.view_vehicle_info(row, col)

elif choice == 4:
    plate_number = input("Enter Vehicle Plate Number to Search:\n>")
    parking_lot.view_vehicle_by_plate(plate_number)

elif choice == 5:
    parking_lot.display_parking_lot()
    print("SPOTS AVAILABLE:", parking_lot.available_spots)

elif choice == 6:
    print("Exiting...")
```

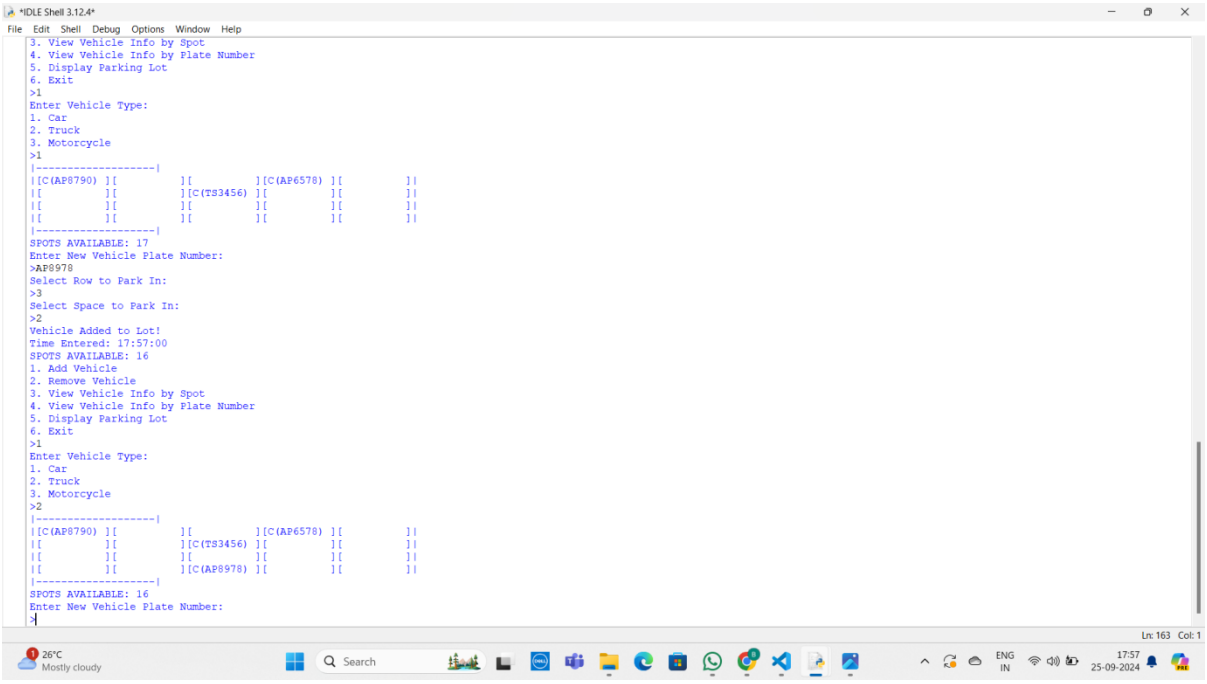
```
            break

        else:
            print("Invalid choice.")

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

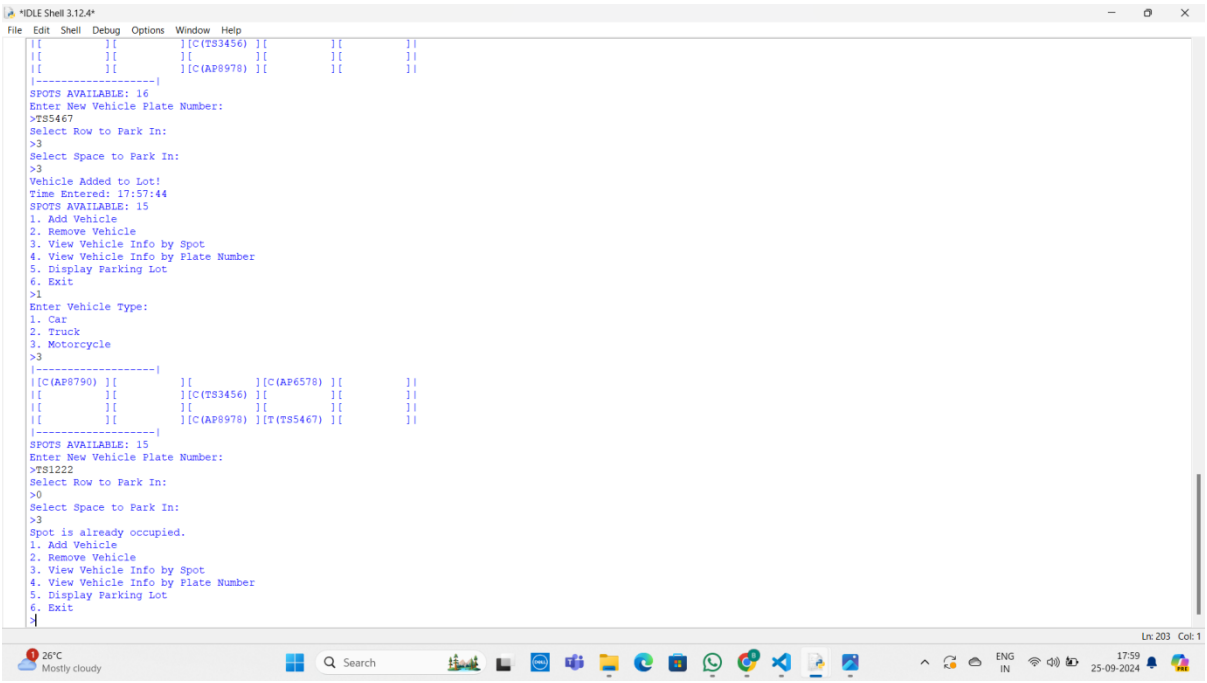
4.RESULT:

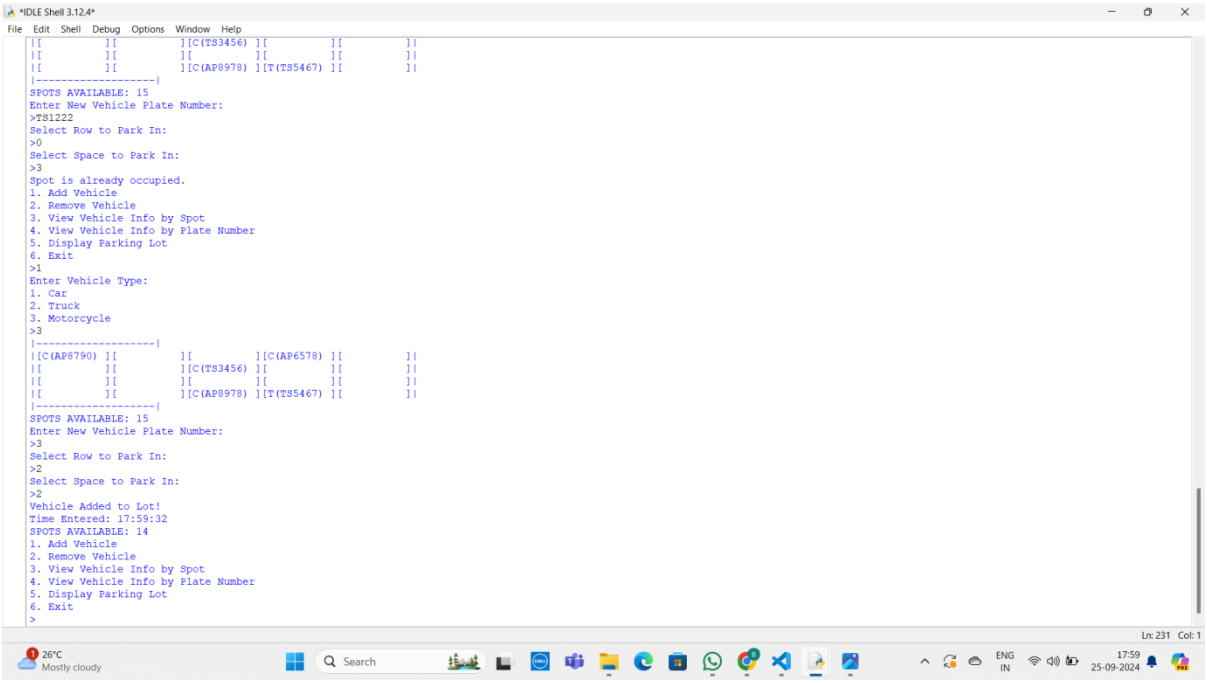

IDLE Shell 3.12.4
File Edit Shell Debug Options Window Help
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/bhavy/OneDrive/Desktop/lot.py
1. Add Vehicle
2. Remove Vehicle
3. View Vehicle Info by Spot
4. View Vehicle Info by Plate Number
5. Display Parking Lot
6. Exit
>1
Enter Vehicle Type:
1. Car
2. Truck
3. Motorcycle
>1
|------------------|
|[ ][ ][ ][ ][ ]|
|[ ][ ][ ][ ][ ]|
|[ ][ ][ ][ ][ ]|
[ ][ ][ ][ ][ ]
SPOTS AVAILABLE: 20
Enter New Vehicle Plate Number:
>AP8790
Select Row to Park In:
>0
Select Space to Park In:
>0
Vehicle Added to Lot!
Time Entered: 17:54:37
SPOTS AVAILABLE: 19
1. Add Vehicle
2. Remove Vehicle
3. View Vehicle Info by Spot
4. View Vehicle Info by Plate Number
5. Display Parking Lot
6. Exit
>1
Enter Vehicle Type:
1. Car
2. Truck
3. Motorcycle
>1
Ln: 8 Col: 36
26°C
Mostly cloudy
Search
ENG
IN
17:54
25-09-2024

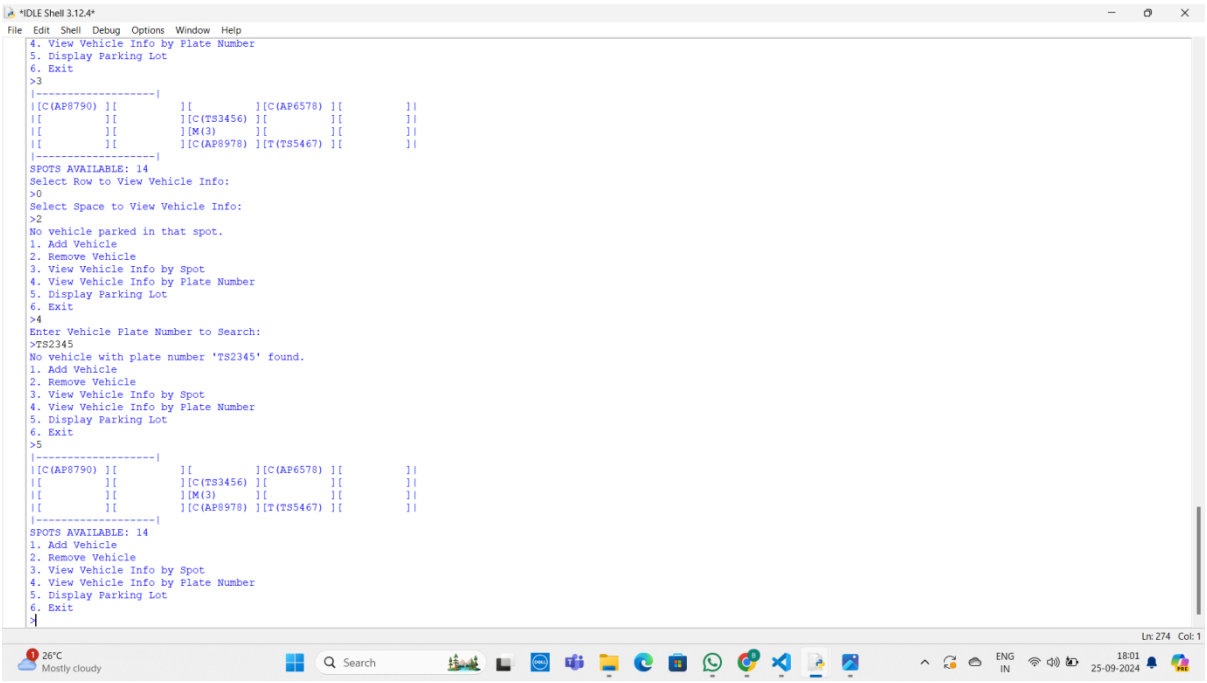

IDLE Shell 3.12.4
File Edit Shell Debug Options Window Help
5. Display Parking Lot
6. Exit
>1
Enter Vehicle Type:
1. Car
2. Truck
3. Motorcycle
>1
|------------------|
|[C(AP8790)][ ][ ][ ][ ]|
|[ ][ ][ ][ ][ ]|
|[ ][ ][ ][ ][ ]|
[ ][ ][ ][ ][ ]
SPOTS AVAILABLE: 19
Enter New Vehicle Plate Number:
>AP6578
Select Row to Park In:
>0
Select Space to Park In:
>3
Vehicle Added to Lot!
Time Entered: 17:55:09
SPOTS AVAILABLE: 18
1. Add Vehicle
2. Remove Vehicle
3. View Vehicle Info by Spot
4. View Vehicle Info by Plate Number
5. Display Parking Lot
6. Exit
>1
Enter Vehicle Type:
1. Car
2. Truck
3. Motorcycle
>1
|------------------|
|[C(AP8790)][ ][ ][C(AP6578)][ ]|
|[ ][ ][ ][ ][ ]|
|[ ][ ][ ][ ][ ]|
[ ][ ][ ][ ][ ]
SPOTS AVAILABLE: 18
Enter New Vehicle Plate Number:
>TS3456
Select Row to Park In:
>
Ln: 83 Col: 1
26°C
Mostly cloudy
Search
ENG
IN
17:56
25-09-2024

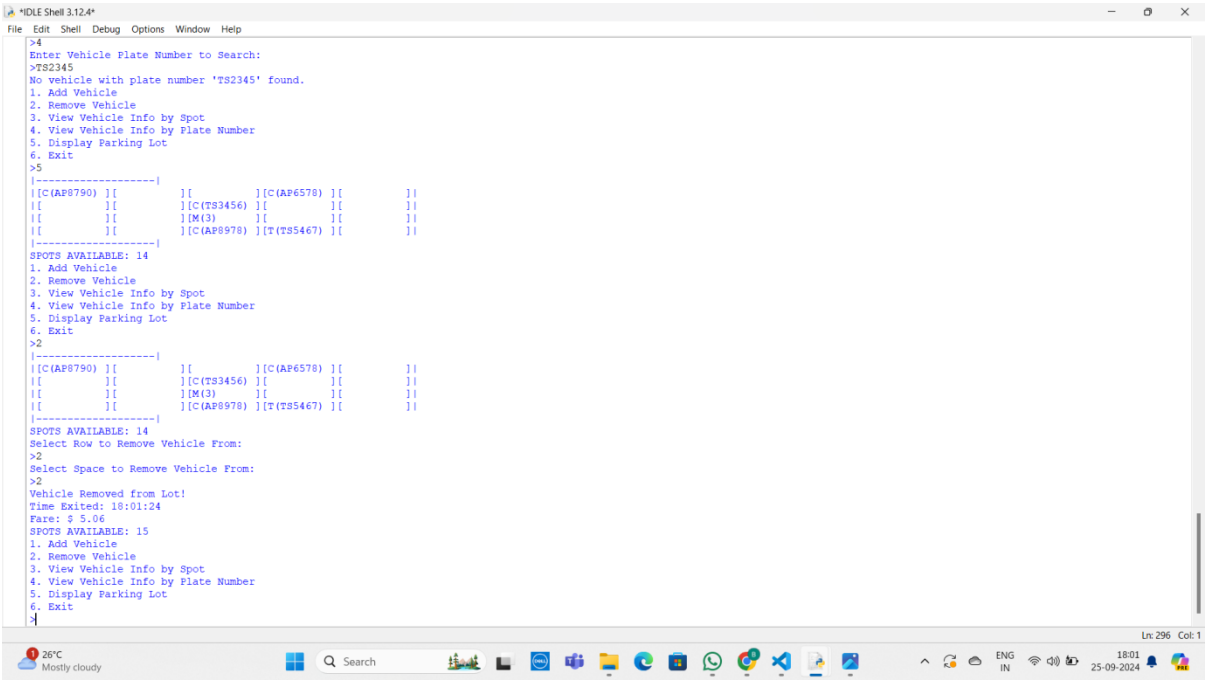

IDLE Shell 3.12.4
File Edit Shell Debug Options Window Help
|[ ][ ][ ][ ][ ]|
|[ ][ ][ ][ ][ ]|
[ ][ ][ ][ ][ ]
SPOTS AVAILABLE: 18
Enter New Vehicle Plate Number:
>TS3456
Select Row to Park In:
>1
Select Space to Park In:
>2
Vehicle Added to Lot!
Time Entered: 17:56:14
SPOTS AVAILABLE: 17
1. Add Vehicle
2. Remove Vehicle
3. View Vehicle Info by Spot
4. View Vehicle Info by Plate Number
5. Display Parking Lot
6. Exit
>1
Enter Vehicle Type:
1. Car
2. Truck
3. Motorcycle
>2
|------------------|
|[C(AP8790)][ ][ ][C(AP6578)][ ]|
|[ ][ ][C(TS3456)][ ][ ]|
|[ ][ ][ ][ ][ ]|
[ ][ ][ ][ ][ ]
SPOTS AVAILABLE: 17
Enter New Vehicle Plate Number:
>TS7856
Select Row to Park In:
>0
Select Space to Park In:
>0
Spot is already occupied.
1. Add Vehicle
2. Remove Vehicle
3. View Vehicle Info by Spot
4. View Vehicle Info by Plate Number
5. Display Parking Lot
6. Exit
>1
Ln: 121 Col: 2
26°C
Mostly cloudy
Search
ENG
IN
17:56
25-09-2024

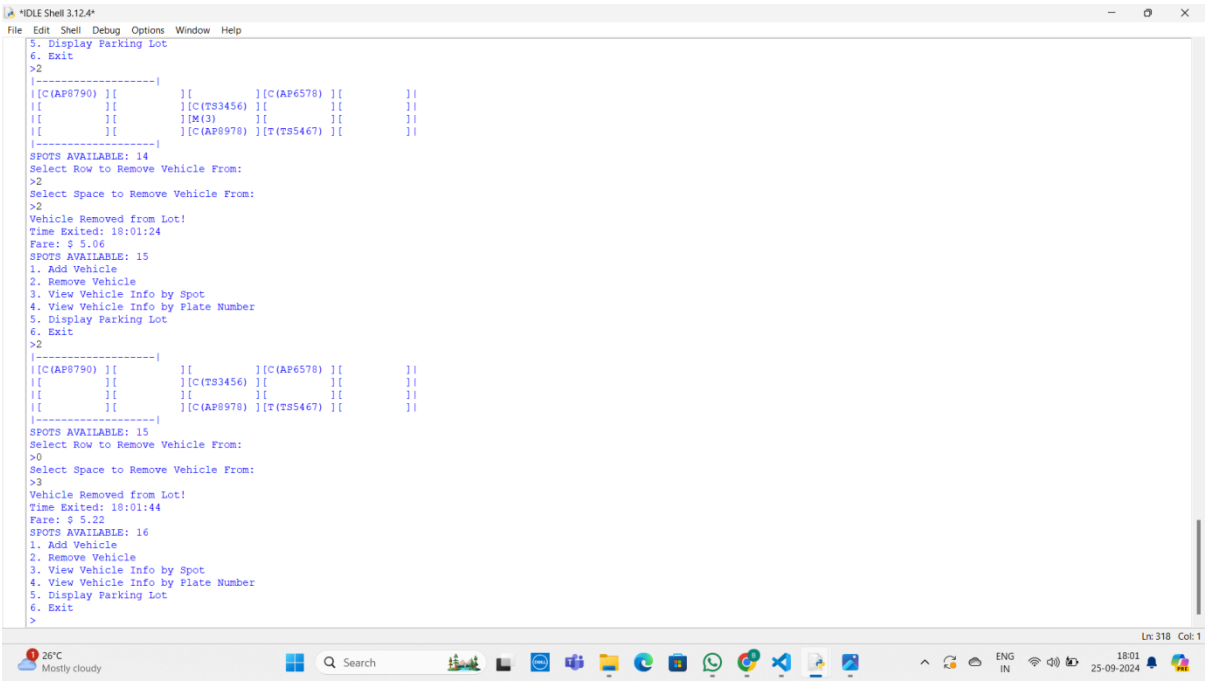

IDLE Shell 3.12.4
File Edit Shell Debug Options Window Help
3. View Vehicle Info by Spot
4. View Vehicle Info by Plate Number
5. Display Parking Lot
6. Exit
>1
Enter Vehicle Type:
1. Car
2. Truck
3. Motorcycle
>1
|------------------|
|[C(AP8790)][ ][ ][C(AP6578)][ ]|
|[ ][ ][C(TS3456)][ ][ ]|
|[ ][ ][ ][ ][ ]|
[ ][ ][ ][ ][ ]
SPOTS AVAILABLE: 17
Enter New Vehicle Plate Number:
>AP8978
Select Row to Park In:
>3
Select Space to Park In:
>2
Vehicle Added to Lot!
Time Entered: 17:57:00
SPOTS AVAILABLE: 16
1. Add Vehicle
2. Remove Vehicle
3. View Vehicle Info by Spot
4. View Vehicle Info by Plate Number
5. Display Parking Lot
6. Exit
>1
Enter Vehicle Type:
1. Car
2. Truck
3. Motorcycle
>2
|------------------|
|[C(AP8790)][ ][ ][C(AP6578)][ ]|
|[ ][ ][C(TS3456)][ ][ ]|
|[ ][ ][ ][ ][ ]|
[ ][ ][C(AP8978)][ ][ ]
SPOTS AVAILABLE: 16
Enter New Vehicle Plate Number:
>
Ln: 163 Col: 1
26°C
Mostly cloudy
Search
ENG
IN
17:57
25-09-2024

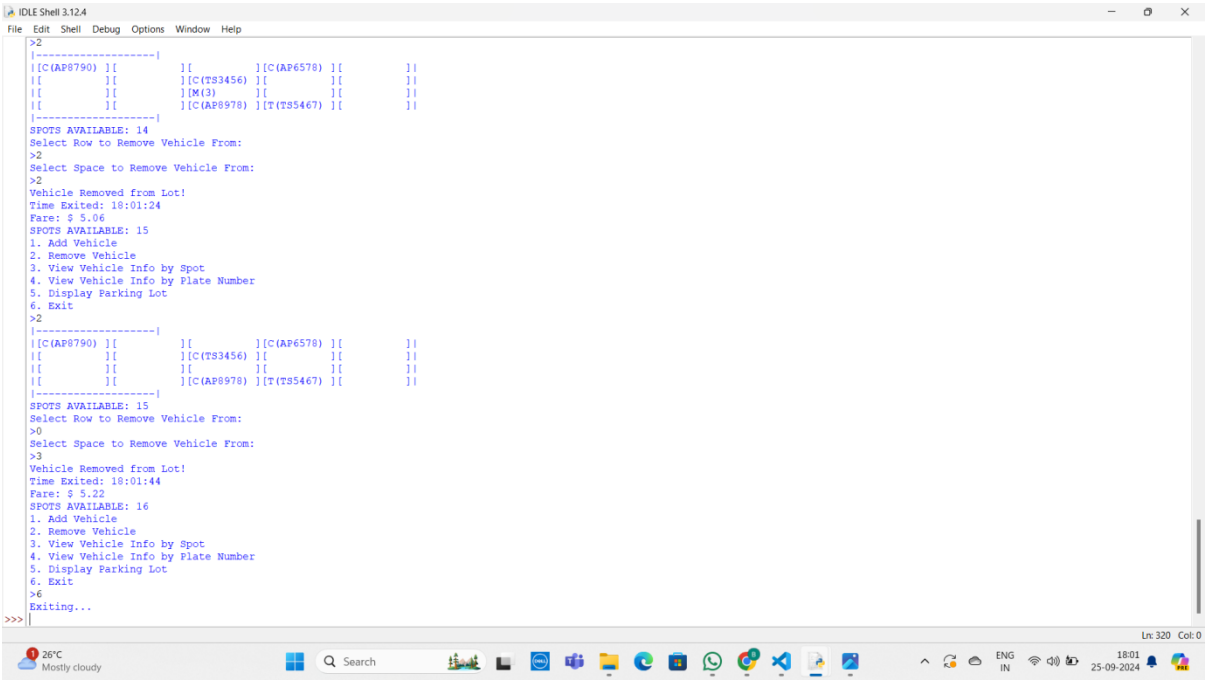

IDLE Shell 3.12.4
File Edit Shell Debug Options Window Help
|[ ][ ][C(TS3456)][ ][ ]|
|[ ][ ][ ][ ][ ]|
[ ][ ][C(AP8978)][ ][ ]
SPOTS AVAILABLE: 16
Enter New Vehicle Plate Number:
>TS5467
Select Row to Park In:
>3
Select Space to Park In:
>3
Vehicle Added to Lot!
Time Entered: 17:57:44
SPOTS AVAILABLE: 15
1. Add Vehicle
2. Remove Vehicle
3. View Vehicle Info by Spot
4. View Vehicle Info by Plate Number
5. Display Parking Lot
6. Exit
>1
Enter Vehicle Type:
1. Car
2. Truck
3. Motorcycle
>3
|------------------|
|[C(AP8790)][ ][ ][C(AP6578)][ ]|
|[ ][ ][C(TS3456)][ ][ ]|
|[ ][ ][ ][ ][ ]|
[ ][ ][C(AP8978)][T(TS5467)][ ]
SPOTS AVAILABLE: 15
Enter New Vehicle Plate Number:
>TS1222
Select Row to Park In:
>0
Select Space to Park In:
>3
Spot is already occupied.
1. Add Vehicle
2. Remove Vehicle
3. View Vehicle Info by Spot
4. View Vehicle Info by Plate Number
5. Display Parking Lot
6. Exit
>
Ln: 203 Col: 1
26°C Mostly cloudy
Search
ENG IN
17:59 25-09-2024


IDLE Shell 3.12.4
File Edit Shell Debug Options Window Help
|[ ][ ][C(TS3456)][ ][ ]|
|[ ][ ][ ][ ][ ]|
[ ][ ][C(AP8978)][T(TS5467)][ ]
SPOTS AVAILABLE: 15
Enter New Vehicle Plate Number:
>TS1222
Select Row to Park In:
>0
Select Space to Park In:
>3
Spot is already occupied.
1. Add Vehicle
2. Remove Vehicle
3. View Vehicle Info by Spot
4. View Vehicle Info by Plate Number
5. Display Parking Lot
6. Exit
>1
Enter Vehicle Type:
1. Car
2. Truck
3. Motorcycle
>3
|------------------|
|[C(AP8790)][ ][ ][C(AP6578)][ ]|
|[ ][ ][C(TS3456)][ ][ ]|
|[ ][ ][ ][ ][ ]|
[ ][ ][C(AP8978)][T(TS5467)][ ]
SPOTS AVAILABLE: 15
Enter New Vehicle Plate Number:
>3
Select Row to Park In:
>2
Select Space to Park In:
>2
Vehicle Added to Lot!
Time Entered: 17:59:32
SPOTS AVAILABLE: 14
1. Add Vehicle
2. Remove Vehicle
3. View Vehicle Info by Spot
4. View Vehicle Info by Plate Number
5. Display Parking Lot
6. Exit
>
Ln: 231 Col: 1
26°C Mostly cloudy
Search
ENG IN
17:59 25-09-2024


IDLE Shell 3.12.4
File Edit Shell Debug Options Window Help
4. View Vehicle Info by Plate Number
5. Display Parking Lot
6. Exit
>3
|--------------------|
|[C(AP8790)][ ][ ][C(AP6578)][ ]|
|[ ][ ][C(TS3456)][ ][ ]|
|[ ][ ][M(3)][ ][ ]|
[ ][ ][C(AP8978)][T(TS5467)][ ]
SPOTS AVAILABLE: 14
Select Row to View Vehicle Info:
>0
Select Space to View Vehicle Info:
>2
No vehicle parked in that spot.
1. Add Vehicle
2. Remove Vehicle
3. View Vehicle Info by Spot
4. View Vehicle Info by Plate Number
5. Display Parking Lot
6. Exit
>4
Enter Vehicle Plate Number to Search:
>TS2345
No vehicle with plate number 'TS2345' found.
1. Add Vehicle
2. Remove Vehicle
3. View Vehicle Info by Spot
4. View Vehicle Info by Plate Number
5. Display Parking Lot
6. Exit
>5
|--------------------|
|[C(AP8790)][ ][ ][C(AP6578)][ ]|
|[ ][ ][C(TS3456)][ ][ ]|
|[ ][ ][M(3)][ ][ ]|
[ ][ ][C(AP8978)][T(TS5467)][ ]
SPOTS AVAILABLE: 14
1. Add Vehicle
2. Remove Vehicle
3. View Vehicle Info by Spot
4. View Vehicle Info by Plate Number
5. Display Parking Lot
6. Exit
>
Ln: 274 Col: 1
26°C Mostly cloudy
Search
ENG IN
18:01 25-09-2024


IDLE Shell 3.12.4
File Edit Shell Debug Options Window Help
>4
Enter Vehicle Plate Number to Search:
>TS2345
No vehicle with plate number 'TS2345' found.
1. Add Vehicle
2. Remove Vehicle
3. View Vehicle Info by Spot
4. View Vehicle Info by Plate Number
5. Display Parking Lot
6. Exit
>5
|--------------------|
|[C(AP8790)][ ][ ][C(AP6578)][ ]|
|[ ][ ][C(TS3456)][ ][ ]|
|[ ][ ][M(3)][ ][ ]|
[ ][ ][C(AP8978)][T(TS5467)][ ]
SPOTS AVAILABLE: 14
1. Add Vehicle
2. Remove Vehicle
3. View Vehicle Info by Spot
4. View Vehicle Info by Plate Number
5. Display Parking Lot
6. Exit
>2
|--------------------|
|[C(AP8790)][ ][ ][C(AP6578)][ ]|
|[ ][ ][C(TS3456)][ ][ ]|
|[ ][ ][M(3)][ ][ ]|
[ ][ ][C(AP8978)][T(TS5467)][ ]
SPOTS AVAILABLE: 14
Select Row to Remove Vehicle From:
>2
Select Space to Remove Vehicle From:
>2
Vehicle Removed from Lot!
Time Exited: 18:01:24
Fare: $ 5.06
SPOTS AVAILABLE: 15
1. Add Vehicle
2. Remove Vehicle
3. View Vehicle Info by Spot
4. View Vehicle Info by Plate Number
5. Display Parking Lot
6. Exit
>
Ln: 296 Col: 1
26°C Mostly cloudy
Search
ENG IN
18:01 25-09-2024


IDLE Shell 3.12.4
File Edit Shell Debug Options Window Help
5. Display Parking Lot
6. Exit
>2
|------------------|
|[C(AP8790)][ ][ ][C(AP6578)][ ]|
|[ ][ ][C(TS3456)][ ][ ]|
|[ ][ ][M(3)][ ][ ]|
[ ][ ][C(AP8978)][T(TS5467)][ ]
SPOTS AVAILABLE: 14
Select Row to Remove Vehicle From:
>2
Select Space to Remove Vehicle From:
>2
Vehicle Removed from Lot!
Time Exited: 18:01:24
Fare: $ 5.06
SPOTS AVAILABLE: 15
1. Add Vehicle
2. Remove Vehicle
3. View Vehicle Info by Spot
4. View Vehicle Info by Plate Number
5. Display Parking Lot
6. Exit
>2
|------------------|
|[C(AP8790)][ ][ ][C(AP6578)][ ]|
|[ ][ ][C(TS3456)][ ][ ]|
|[ ][ ][ ][ ][ ]|
[ ][ ][C(AP8978)][T(TS5467)][ ]
SPOTS AVAILABLE: 15
Select Row to Remove Vehicle From:
>0
Select Space to Remove Vehicle From:
>3
Vehicle Removed from Lot!
Time Exited: 18:01:44
Fare: $ 5.22
SPOTS AVAILABLE: 16
1. Add Vehicle
2. Remove Vehicle
3. View Vehicle Info by Spot
4. View Vehicle Info by Plate Number
5. Display Parking Lot
6. Exit
>
Ln: 318 Col: 1
26°C Mostly cloudy
Search
ENG IN
18:01 25-09-2024


IDLE Shell 3.12.4
File Edit Shell Debug Options Window Help
>2
|------------------|
|[C(AP8790)][ ][ ][C(AP6578)][ ]|
|[ ][ ][C(TS3456)][ ][ ]|
|[ ][ ][M(3)][ ][ ]|
[ ][ ][C(AP8978)][T(TS5467)][ ]
SPOTS AVAILABLE: 14
Select Row to Remove Vehicle From:
>2
Select Space to Remove Vehicle From:
>2
Vehicle Removed from Lot!
Time Exited: 18:01:24
Fare: $ 5.06
SPOTS AVAILABLE: 15
1. Add Vehicle
2. Remove Vehicle
3. View Vehicle Info by Spot
4. View Vehicle Info by Plate Number
5. Display Parking Lot
6. Exit
>2
|------------------|
|[C(AP8790)][ ][ ][C(AP6578)][ ]|
|[ ][ ][C(TS3456)][ ][ ]|
|[ ][ ][ ][ ][ ]|
[ ][ ][C(AP8978)][T(TS5467)][ ]
SPOTS AVAILABLE: 15
Select Row to Remove Vehicle From:
>0
Select Space to Remove Vehicle From:
>3
Vehicle Removed from Lot!
Time Exited: 18:01:44
Fare: $ 5.22
SPOTS AVAILABLE: 16
1. Add Vehicle
2. Remove Vehicle
3. View Vehicle Info by Spot
4. View Vehicle Info by Plate Number
5. Display Parking Lot
6. Exit
>6
Exiting...
>>>
Ln: 320 Col: 0
26°C Mostly cloudy
Search
ENG IN
18:01 25-09-2024

5.CONCLUSION:

The Parking Lot Management System represents a significant step forward in the management of parking facilities. By integrating modern technology and user-centric design, it not only simplifies operations but also enhances user satisfaction. Whether you are a small business owner or managing a large parking facility, this system equips you with the tools necessary to optimize your operations and improve the parking experience.

Explore the functionalities of this innovative solution and discover how it can transform your parking management approach!

6.REFERENCES:

- Gemini AI
- ChatGPT
- GitHub
- Udemy(paltform)
- Python reference book(By KV Rao)