**PSEUDOCODE**

Class Vehicle:

Constructor Vehicle(vehicle\_number, vehicle\_type, vehicle\_name, owner\_name)

Method park()

Method unpark()

Method calculate\_parking\_charge()

Method display\_details()

Class ParkingLot:

Constructor ParkingLot(capacity)

Method park\_vehicle(vehicle)

Method unpark\_vehicle(vehicle\_number)

Method search\_vehicle(vehicle\_number)

Method display\_parked\_vehicles()

Method display\_parking\_lot\_status()

Function main():

Create a new ParkingLot object with capacity 50

While True:

Display main menu options and get user's choice

If choice is '1':

Get vehicle details from user and park the vehicle

Else if choice is '2':

Get vehicle number from user and unpark the vehicle

Else if choice is '3':

Get vehicle number from user and search for the vehicle

Else if choice is '4':

Display details of all parked vehicles

Else if choice is '5':

Display current status of the parking lot

Else if choice is '6':

Exit the program

Else:

Display "Invalid choice. Please try again."

Call the main function to start the Vehicle Parking Management System

**SOURCE CODE DOCUMENTATION**

Vehicle Parking Management System Documentation

**Introduction**

The Vehicle Parking Management System is a simple Python program that simulates the management of a parking lot. It allows users to park vehicles, track their entry and exit times, calculate parking charges, and manage the parking lot's occupancy.

**Classes**

**Vehicle Class**

The Vehicle class represents a single vehicle that can be parked in the parking lot.

**Attributes**

* vehicle\_number: The vehicle's number plate (str).
* vehicle\_type: The type of the vehicle (e.g., Car, Motorcycle, Bicycle) (str).
* vehicle\_name: The name of the vehicle (str).
* owner\_name: The name of the vehicle owner (str).
* entry\_time: The timestamp when the vehicle is parked (datetime).
* exit\_time: The timestamp when the vehicle is unparked (datetime).
* parking\_duration: The duration of parking (timedelta).
* parking\_charge: The calculated parking charge (float).
* payment\_status: The status of payment (Paid: True, Pending: False) (bool).

**Methods**

* \_\_init\_\_(self, vehicle\_number, vehicle\_type, vehicle\_name, owner\_name): Initializes the Vehicle object with provided details.
* park(self): Parks the vehicle and records the entry time.
* unpark(self): Unparks the vehicle, calculates the parking charge, and updates the payment status.
* calculate\_parking\_charge(self): Calculates the parking charge based on the parking duration and vehicle type.
* display\_details(self): Displays the vehicle details including entry time, exit time, parking duration, parking charge, and payment status.

**ParkingLot Class**

The ParkingLot class represents the parking lot and its operations.

**Attributes**

* capacity: The total number of parking spaces in the parking lot (int).
* available\_spaces: The number of available parking spaces (int).
* occupied\_spaces: The number of occupied parking spaces (int).
* vehicles: A dictionary that stores parked vehicles (vehicle\_number: Vehicle object).

**Methods**

* \_\_init\_\_(self, capacity): Initializes the ParkingLot object with the given capacity.
* park\_vehicle(self, vehicle): Parks the vehicle in the parking lot if space is available.
* unpark\_vehicle(self, vehicle\_number): Unparks the vehicle from the parking lot.
* search\_vehicle(self, vehicle\_number): Searches for a vehicle in the parking lot and displays its details if found.
* display\_parked\_vehicles(self): Displays the details of all parked vehicles in the parking lot.
* display\_parking\_lot\_status(self): Displays the current status of the parking lot including total spaces, occupied spaces, and available spaces.

**Main Function**

The main() function is the entry point of the program. It creates a ParkingLot object with a specified capacity and provides a menu-driven interface for users to interact with the system.

**How to Use**

To use the Vehicle Parking Management System:

Run the Python script.

Choose an option from the displayed menu:

Enter 1 to park a vehicle and provide vehicle details.

Enter 2 to unpark a vehicle and provide the vehicle number.

Enter 3 to search for a vehicle and provide the vehicle number.

Enter 4 to display details of all parked vehicles.

Enter 5 to display the current parking lot status.

Enter 6 to exit the program.

**Example**

Suppose a user chooses to park a car with the following details:

Vehicle Number: KBC 1234j

Vehicle Type: Car

Vehicle Name: Toyota

Owner Name: Claire Simiyu

The system will display the entry time, and later, when the vehicle is unparked, it will show the exit time, parking duration, parking charge, and payment status.

**Assumptions and Limitations**

* The parking charges are calculated based on the vehicle type (Car, Motorcycle, or other) and parking duration.
* The system does not include features for vehicle registration or payment processing.
* The user can park and unpark vehicles by providing the vehicle number, but there is no additional security mechanism for verification.
* The system assumes that vehicles will be parked and unparked in sequential order without overlapping entries/exits.

**SCREENSHOTS**





