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()
```