Fundamentals of Programming II

Final Project Report:

Mtag Management System

Name: Rumman Tanvir

CMS: 482064

Department of Aerospace Engineering
School of Mechanical and Manufacturing Engineering
National University of Science and Technology

INTRODUCTION

The MTag System project is a system created to make toll management easier for vehicles traveling between cities. The system is inspired by the real-world need for more efficient toll collection. It allows users to register vehicles, get assigned unique MTags, manage balances, and calculate toll charges at different toll plazas. By automating these tasks, the system provides a smoother and quicker way to handle toll payments, making it simpler for both vehicle owners and toll authorities.

CODE

Importing libraries

```
import pandas as pd
import random
```

The Pandas library is used for working with data structures like DataFrames and Series. It helps handle data presented in columns or tables. The random module generates random numbers, which is used in this code to assign a unique Mtag number to a registered vehicle.

Declaring Dataframes

```
# Dataframe to store the vehicle information
vehicles = pd.DataFrame(columns=["Mtag Number", "CNIC", "Number Plate", "Owner",
"Contact", "Balance", "City"])

# Dataframe to store toll information
toll_info = pd.DataFrame(columns=["City", "Toll Fee"])
```

Two data frames are created to store the details about the registered vehicles and the toll info of the cities. the pd.DataFrame command creates an empty Dataframe with columns to specify the structure. They are initially empty because the data will be added once the code is running.

Saving and loading data to and from CSV files

```
# Function to save vehicles data to a CSV file
def save_to_excel():
  vehicles.to_csv(r"C:\Programming - Pycharm\MTAGG.csv", index=False)  # Use raw string for
the file path
  toll_info.to_csv(r"C:\Programming - Pycharm\TollInfo.csv", index=False)
  print("Vehicles data and toll info have been saved.")


# Function to load vehicles data from CSV file
def load_from_excel():
  try:
    vehicles_data = pd.read_csv(r"C:\Programming - Pycharm\MTAGG.csv")  # Use raw string for
the file path
    toll_data = pd.read_csv(r"C:\Programming - Pycharm\TollInfo.csv")
    return vehicles_data, toll_data
  except FileNotFoundError:
    print("No previous data found.")
    return pd.DataFrame(
      columns=["Mtag Number", "CNIC", "Number Plate", "Owner", "Contact", "Balance",
"City"]), pd.DataFrame(
      columns=["City", "Toll Fee"])
```

Two functions are created, one to save the data to a CSV file and one to load data. It saves the two data frames *vehicles* and *toll_info* so that once the program is closed, the data is saved, otherwise all changes would be lost when the program exits

For the loading data function, an additional command is added to indicate whether any previous data exists in the excel file or not. The *try-except* command handles any errors, so that if the file is missing the program won't crash. The *FileNotFoundError* is a specific error that's raised when the program can't locate the file

The location of the file path is the same location as the program.

Registering a vehicle

```
# Registering a new vehicle
def register_vehicle(cnic, number_plate=None, car_model=None, owner_name=None,
contact=None, city=None):
  if cnic in vehicles["CNIC"].values:
    print("This CNIC is already registered.")
    choice = input("Do you want to register another vehicle under the same CNIC? (yes/no):
").lower()
    if choice == "yes":
      owner_name = vehicles.loc[vehicles["CNIC"] == cnic, "Owner"].values[0]
      contact = vehicles.loc[vehicles["CNIC"] == cnic, "Contact"].values[0]
      city = vehicles.loc[vehicles["CNIC"] == cnic, "City"].values[0]
    else:
      print("No additional vehicle registered.")
```

```
        return
    else:
        owner_name = owner_name or input("Enter Owner Name: ")
        contact = contact or input("Enter Contact Number (XXXX-XXXXXXX): ")
        city = city or input("Enter City: ")

    number_plate = number_plate or input("Enter Number Plate: ")
    car_model = car_model or input("Enter Car Model: ")

    mtag_number = random.randint(1000, 9999)
    new_vehicle = {
        "Mtag Number": mtag_number,
        "CNIC": cnic,
        "Number Plate": number_plate,
        "Car Model": car_model,
        "Owner": owner_name,
        "Contact": contact,
        "Balance": 0,
        "City": city,
    }
    vehicles.loc[len(vehicles)] = new_vehicle
    print(f"Vehicle with CNIC {cnic} has been registered successfully.")
    print(f"Vehicle Mtag Number: {mtag_number}")
```

This function is used to register a new vehicle. It adds a new vehicle to the vehicles DataFrame and checks to see if the CNIC is already registered. It allows multiple vehicles to be registered under the same CNIC, and assigns a unique Mtag to each vehicle. The *vehicles.loc* command adds the data as a new row in the vehicles DataFrame

Adding city toll and displaying all vehicle information

```
# Function to add toll information based on city
def add_toll_info(city, toll_fee):
    if city not in toll_info["City"].values:
        new_toll = {"City": city, "Toll Fee": toll_fee}
        toll_info.loc[len(toll_info)] = new_toll
        print(f"Toll fee for {city} has been added with fee ${toll_fee}.")
    else:
        print(f"Toll fee for {city} already exists.")

# Displaying all vehicles that are registered
def display_vehicles():
    if vehicles.empty:
```

```
    print("No vehicles registered.")
  else:
    print(vehicles)
```

These two functions are used to add the toll information of a city and to display any registered vehicles. The *toll_info["City"].values* command extracts all city names into an array to check if the city already exists. If not, *loc[len(toll_info)] = new_toll* appends a new row by targeting the next index (using len). The dictionaries *({"City": city, "Toll Fee": toll_fee*}) are used to represent a single row. It also avoids duplicates by checking if the city is already present in *toll_*info. The second function is used to display all the vehicles and their respective information.

Adding balance to a vehicle account and displaying vehicle details based on Mtag number

```
# Add balance to a vehicle's account
def add_balance(cnic, mtag_number, amount):
  if cnic in vehicles["CNIC"].values and mtag_number in vehicles["Mtag Number"].values:
    if not vehicles[(vehicles["CNIC"] == cnic) & (vehicles["Mtag Number"] ==
mtag_number)].empty:
      vehicles.loc[(vehicles["CNIC"] == cnic) & (vehicles["Mtag Number"] == mtag_number),
"Balance"] += amount
      new_balance = vehicles.loc[(vehicles["CNIC"] == cnic) & (vehicles["Mtag Number"] ==
mtag_number), "Balance"].values[0]
      print(f"Added ${amount} to vehicle with Mtag Number {mtag_number}. \nNew balance:
${new_balance}.")
    else:
      print("The provided CNIC and Mtag Number do not match.")
  else:
    print("Invalid CNIC or Mtag Number.")


# Find and display vehicle details based on Mtag number
def display_by_mtag(mtag_number):
  vehicle = vehicles[vehicles["Mtag Number"] == mtag_number]
  if not vehicle.empty:
    print("Vehicle details:")
    print(vehicle)
  else:
    print("No vehicle found with the given Mtag number.")
```

The *add_balance* function adds money to a specific vehicle's account. The combination *(vehicles["CNIC"] == cnic) & (vehicles["Mtag Number"] == mtag_number)* filters the DataFrame and allots the amount to that specific Mtag with the registered CNIC. The *vehicles.loc* command updates the row. The second function checks whether an Mtag has been allotted, and if so, display all the vehicle information associated with that Mtag.

Toll deduction

```
# Calculate and deduct toll based on the city
def calculate_and_deduct_toll(cnic, mtag_number, city):
  if cnic in vehicles["CNIC"].values and mtag_number in vehicles["Mtag Number"].values:
    if not vehicles[(vehicles["CNIC"] == cnic) & (vehicles["Mtag Number"] ==
mtag_number)].empty:
      toll_fee = toll_info.loc[toll_info["City"] == city, "Toll Fee"].values
      if toll_fee.size > 0:
        toll_fee = toll_fee[0]
        current_balance = vehicles.loc[(vehicles["CNIC"] == cnic) & (vehicles["Mtag Number"]
== mtag_number), "Balance"].values[0]
        if current_balance >= toll_fee:
          vehicles.loc[(vehicles["CNIC"] == cnic) & (vehicles["Mtag Number"] == mtag_number),
"Balance"] -= toll_fee
          print(f"Toll fee of ${toll_fee} for city {city} has been deducted from vehicle with Mtag
Number {mtag_number}.")
        else:
          print("Insufficient balance for toll deduction.")
      else:
        print(f"No toll fee information found for city {city}.")
    else:
      print("The provided CNIC and Mtag Number do not match.")
  else:
    print("Invalid CNIC or Mtag Number.")
```

This function deducts toll fees for a specific trip. *toll_info.loc[...]* retrieves the toll fee for the city and .values gets the raw data (as a NumPy array). If the balance is sufficient in the account, the toll fee is deducted based on the city.

Main loop

```

# Main code that runs with a while loop until user exits
while True:
  print("\nOptions for Mtag management:")
  print("1. Register a vehicle")
  print("2. Display all vehicles")
```

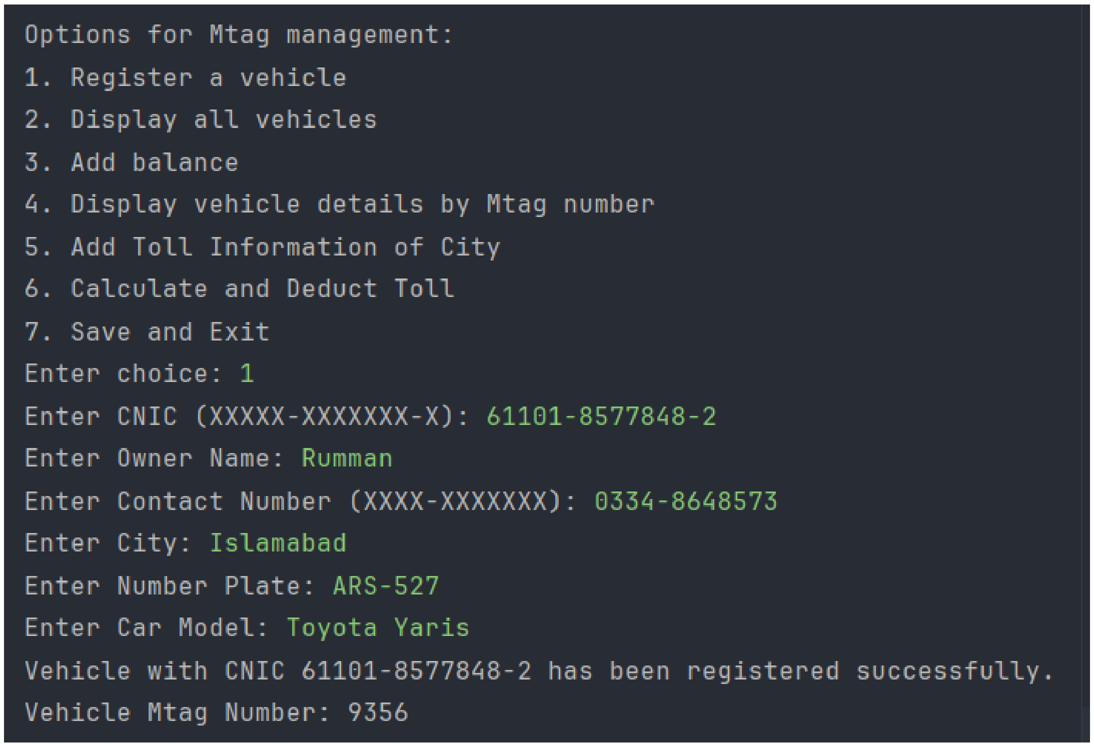
```python
print("3. Add balance")
print("4. Display vehicle details by Mtag number")
print("5. Add Toll Information of City")
print("6. Calculate and Deduct Toll")
print("7. Save and Exit")

choice = input("Enter choice: ")

if choice == "1":
    cnic = input("Enter CNIC (XXXXX-XXXXXXX-X): ")
    register_vehicle(cnic)
elif choice == "2":
    display_vehicles()
elif choice == "3":
    cnic = input("Enter CNIC (XXXXX-XXXXXXX-X): ")
    if cnic in vehicles["CNIC"].values:
        mtag_number = int(input("Enter Mtag Number: "))
        if mtag_number in vehicles.loc[vehicles["CNIC"] == cnic, "Mtag Number"].values:
            amount = float(input("Enter amount to add ($): "))
            add_balance(cnic, mtag_number, amount)
        else:
            print("Invalid Mtag Number. Please try again.")
    else:
        print("Invalid CNIC. The vehicle is not registered.")
elif choice == "4":
    mtag_number = int(input("Enter Mtag Number: "))
    display_by_mtag(mtag_number)
elif choice == "5":
    city = input("Enter the city for the toll information: ")
    toll_fee = float(input("Enter the toll fee for this city: "))
    add_toll_info(city, toll_fee)
elif choice == "6":
    cnic = input("Enter CNIC (XXXXX-XXXXXXX-X): ")
    if cnic in vehicles["CNIC"].values:
        mtag_number = int(input("Enter Mtag Number: "))
        if mtag_number in vehicles.loc[vehicles["CNIC"] == cnic, "Mtag Number"].values:
            city = input("Enter the city name for toll deduction: ")
            calculate_and_deduct_toll(cnic, mtag_number, city)
        else:
            print("Invalid Mtag Number. Please try again.")
    else:
        print("Invalid CNIC. The vehicle is not registered.")
elif choice == "7":
    save_to_excel()
```

```
        print("Exiting system...")
        break
    else:
        print("Invalid choice. Please try again.")
```

The main loop provides a user-friendly menu for the Mtag management system. Each choice maps one of the functions created above based on what the user wants to do. The while loop will continue to run until the user hits option 7, which is to exit the code and save all data in the Excel file.

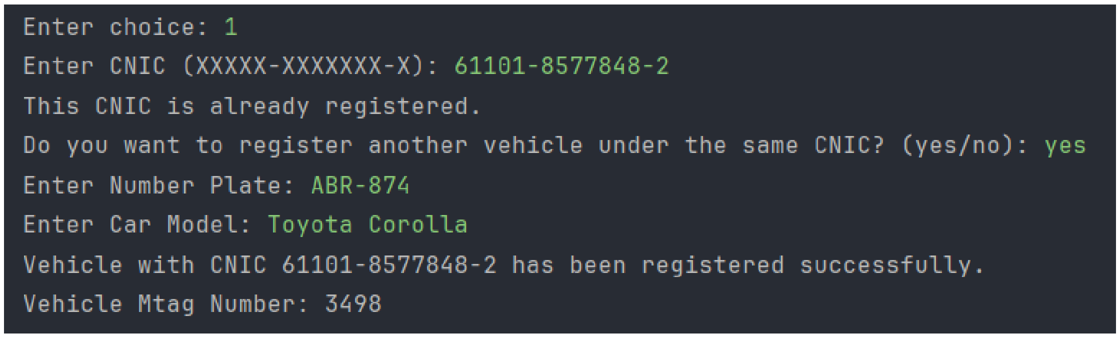
<u>OUTPUT</u>

Option 1: Registering a Vehicle



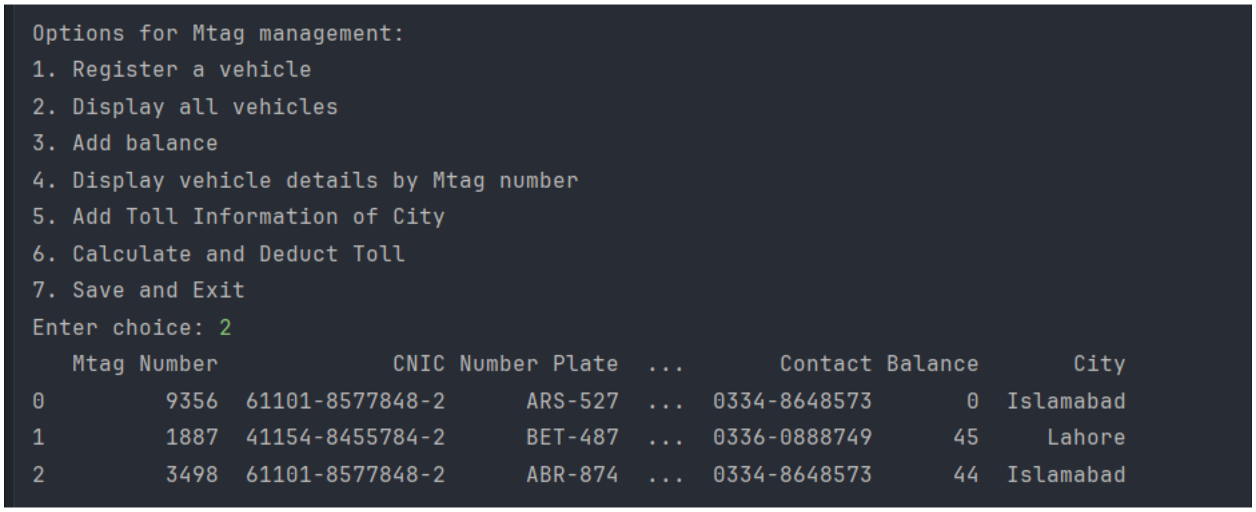
```
Options for Mtag management:
1. Register a vehicle
2. Display all vehicles
3. Add balance
4. Display vehicle details by Mtag number
5. Add Toll Information of City
6. Calculate and Deduct Toll
7. Save and Exit
Enter choice: 1
Enter CNIC (XXXXX-XXXXXXX-X): 61101-8577848-2
Enter Owner Name: Rumman
Enter Contact Number (XXXX-XXXXXXX): 0334-8648573
Enter City: Islamabad
Enter Number Plate: ARS-527
Enter Car Model: Toyota Yaris
Vehicle with CNIC 61101-8577848-2 has been registered successfully.
Vehicle Mtag Number: 9356
```

Registering another vehicle on the same CNIC



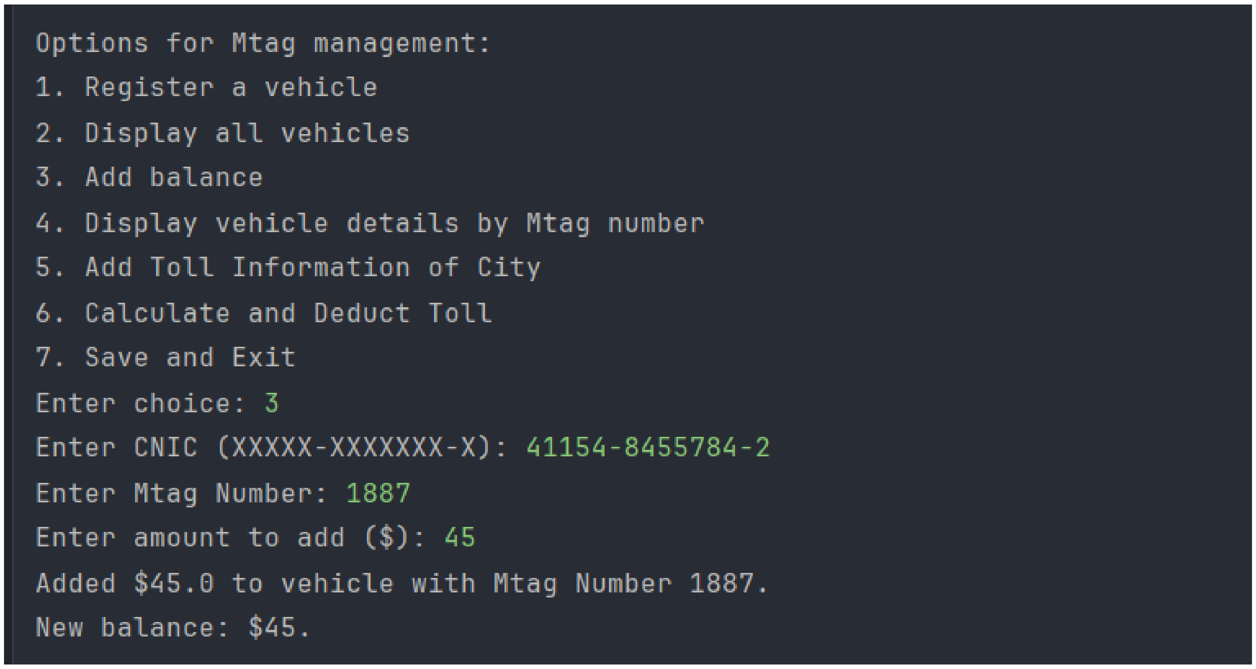
```
Enter choice: 1
Enter CNIC (XXXXX-XXXXXXX-X): 61101-8577848-2
This CNIC is already registered.
Do you want to register another vehicle under the same CNIC? (yes/no): yes
Enter Number Plate: ABR-874
Enter Car Model: Toyota Corolla
Vehicle with CNIC 61101-8577848-2 has been registered successfully.
Vehicle Mtag Number: 3498
```

Option 2: Displaying all Vehicles



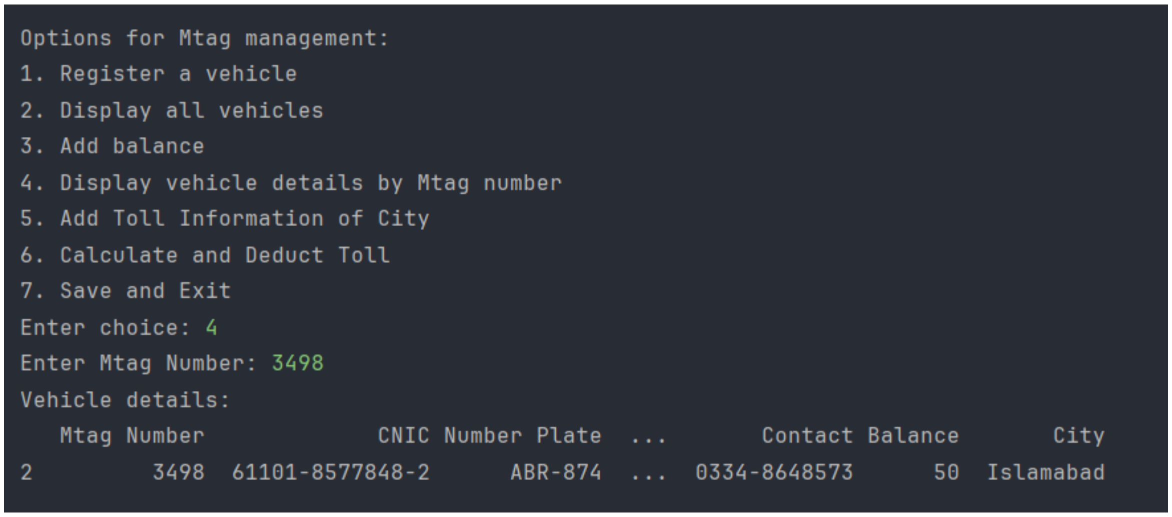
```
Options for Mtag management:
1. Register a vehicle
2. Display all vehicles
3. Add balance
4. Display vehicle details by Mtag number
5. Add Toll Information of City
6. Calculate and Deduct Toll
7. Save and Exit
Enter choice: 2
   Mtag Number             CNIC Number Plate  ...       Contact Balance       City
0         9356  61101-8577848-2      ARS-527  ...  0334-8648573       0  Islamabad
1         1887  41154-8455784-2      BET-487  ...  0336-0888749      45     Lahore
2         3498  61101-8577848-2      ABR-874  ...  0334-8648573      44  Islamabad
```

Option 3: Adding Balance



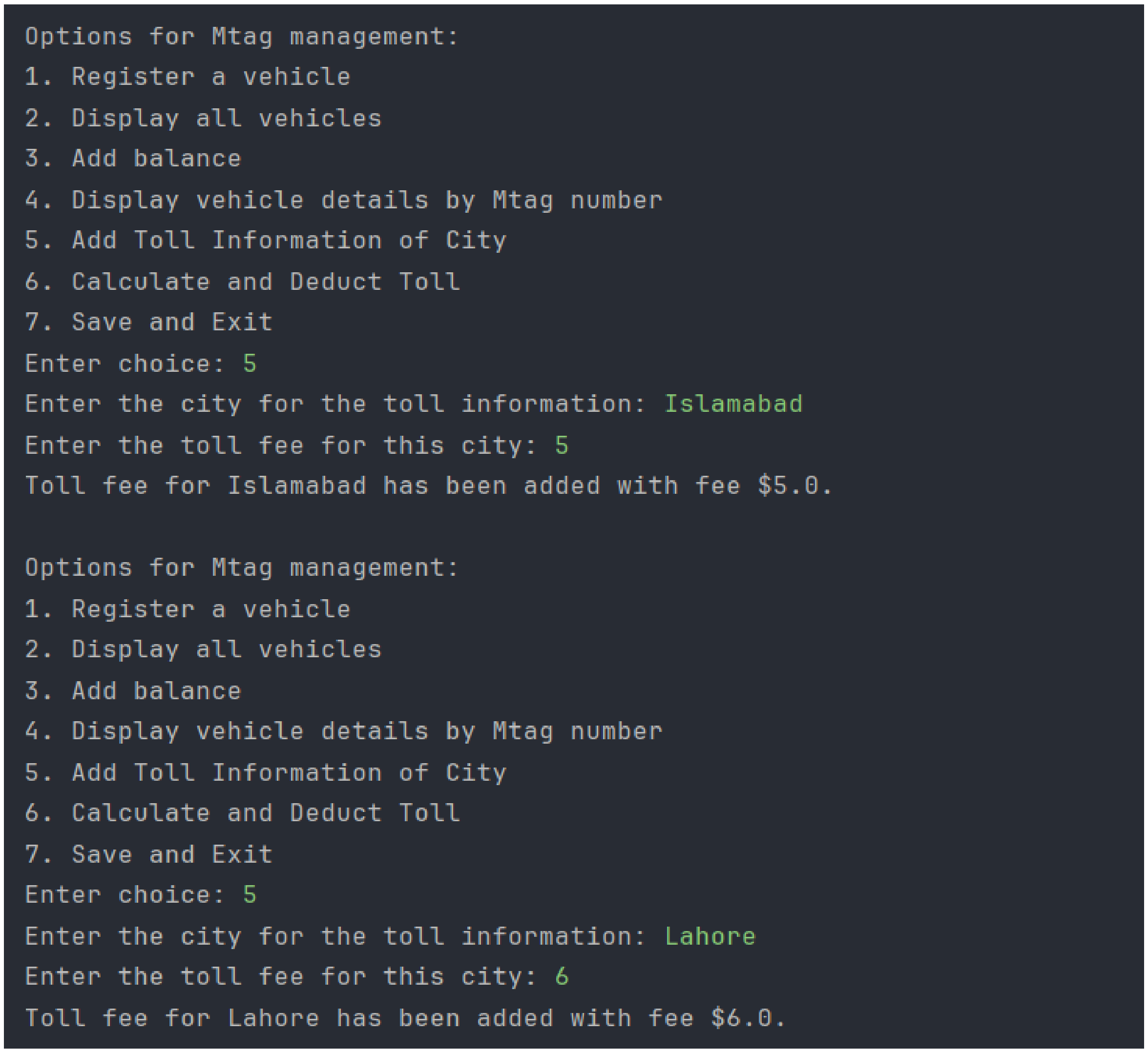
```
Options for Mtag management:
1. Register a vehicle
2. Display all vehicles
3. Add balance
4. Display vehicle details by Mtag number
5. Add Toll Information of City
6. Calculate and Deduct Toll
7. Save and Exit
Enter choice: 3
Enter CNIC (XXXXX-XXXXXXX-X): 41154-8455784-2
Enter Mtag Number: 1887
Enter amount to add ($): 45
Added $45.0 to vehicle with Mtag Number 1887.
New balance: $45.
```

Option 4: Displaying vehicle details by Mtag number



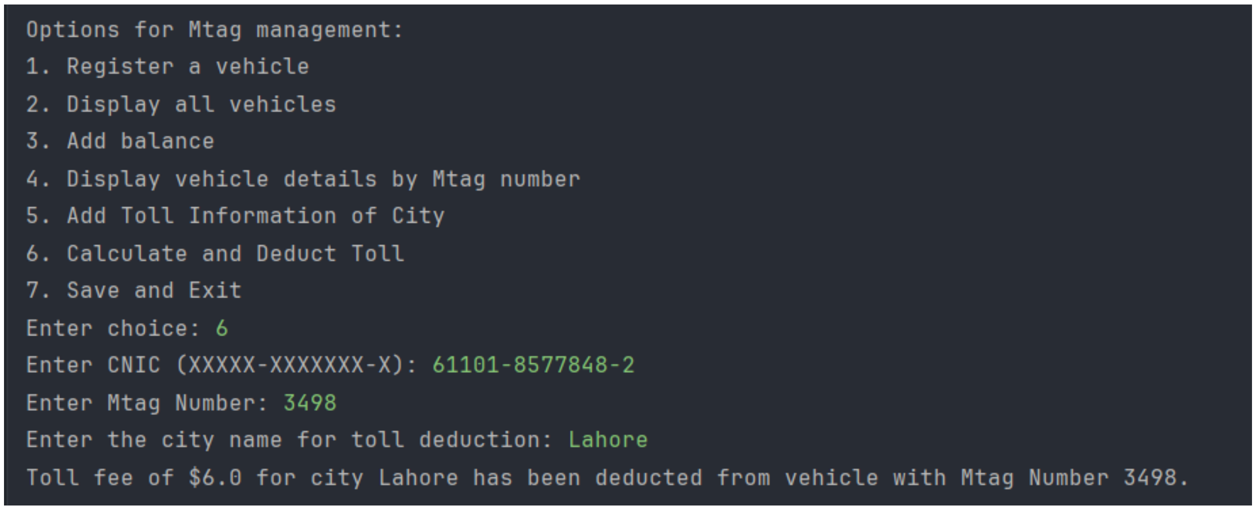
```
Options for Mtag management:
1. Register a vehicle
2. Display all vehicles
3. Add balance
4. Display vehicle details by Mtag number
5. Add Toll Information of City
6. Calculate and Deduct Toll
7. Save and Exit
Enter choice: 4
Enter Mtag Number: 3498
Vehicle details:
   Mtag Number             CNIC Number Plate  ...       Contact Balance       City
2         3498  61101-8577848-2      ABR-874  ...  0334-8648573      50  Islamabad
```

Option 5: Adding Toll information of city



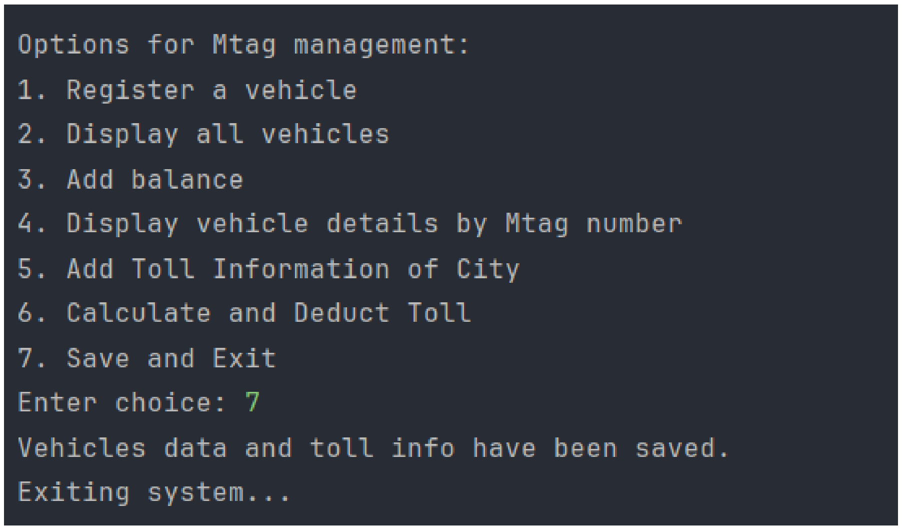
```
Options for Mtag management:
1. Register a vehicle
2. Display all vehicles
3. Add balance
4. Display vehicle details by Mtag number
5. Add Toll Information of City
6. Calculate and Deduct Toll
7. Save and Exit
Enter choice: 5
Enter the city for the toll information: Islamabad
Enter the toll fee for this city: 5
Toll fee for Islamabad has been added with fee $5.0.

Options for Mtag management:
1. Register a vehicle
2. Display all vehicles
3. Add balance
4. Display vehicle details by Mtag number
5. Add Toll Information of City
6. Calculate and Deduct Toll
7. Save and Exit
Enter choice: 5
Enter the city for the toll information: Lahore
Enter the toll fee for this city: 6
Toll fee for Lahore has been added with fee $6.0.
```

Option 6: Deducting Toll



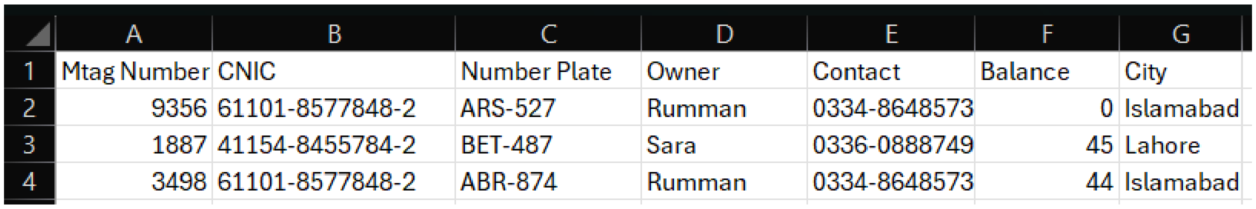
```
Options for Mtag management:
1. Register a vehicle
2. Display all vehicles
3. Add balance
4. Display vehicle details by Mtag number
5. Add Toll Information of City
6. Calculate and Deduct Toll
7. Save and Exit
Enter choice: 6
Enter CNIC (XXXXX-XXXXXXX-X): 61101-8577848-2
Enter Mtag Number: 3498
Enter the city name for toll deduction: Lahore
Toll fee of $6.0 for city Lahore has been deducted from vehicle with Mtag Number 3498.
```

Option 7: Save and Exit

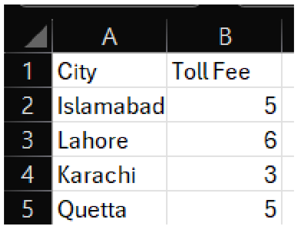


```
Options for Mtag management:
1. Register a vehicle
2. Display all vehicles
3. Add balance
4. Display vehicle details by Mtag number
5. Add Toll Information of City
6. Calculate and Deduct Toll
7. Save and Exit
Enter choice: 7
Vehicles data and toll info have been saved.
Exiting system...
```

Information saved and stored in Excel Spreadsheet in specified file path



	A	B	C	D	E	F	G
1	Mtag Number	CNIC	Number Plate	Owner	Contact	Balance	City
2	9356	61101-8577848-2	ARS-527	Rumman	0334-8648573	0	Islamabad
3	1887	41154-8455784-2	BET-487	Sara	0336-0888749	45	Lahore
4	3498	61101-8577848-2	ABR-874	Rumman	0334-8648573	44	Islamabad



	A	B
1	City	Toll Fee
2	Islamabad	5
3	Lahore	6
4	Karachi	3
5	Quetta	5