Object Oriented Programming (SE-103-22-A)

Traffic Management System

Documentation

Submitted By

Waleed Khattak	**21814198-001**
Rida Ashraf	**22014198-070**
Zain Ul Abdeen	**22014198-078**

Submitted To

Sir Naveed Abbas

Spring 2023

Department of Software Engineering

Traffic Management System

Contents:

Chapter 1: Introduction and planning

Chapter 2: Requirement Engineering

Chapter 3: Implementation

Chapter 4: Deployment and Maintenance

Chapter 5: Conclusion

Chapter 6: Code and Output

Chapter 1: Introduction and Planning

1.1 Introduction

The Traffic Management System (TMS) is a sophisticated software solution designed to enhance the efficiency, accuracy, and management of traffic-related activities and violations. The primary objective of this project is to develop a comprehensive system that automates and streamlines the process of vehicle registration, challan issuance, challan payment, and violation record management. By leveraging modern technology and intuitive interfaces, the TMS aims to contribute to improved traffic management, better compliance with traffic regulations, and enhanced user experience.

1.2 Project Feasibility Report

The project feasibility report assesses the technical, schedule, economic, and operational aspects of the Traffic Management System. It ensures that the proposed system aligns with the organization's goals and meets the expectations of stakeholders and end-users.

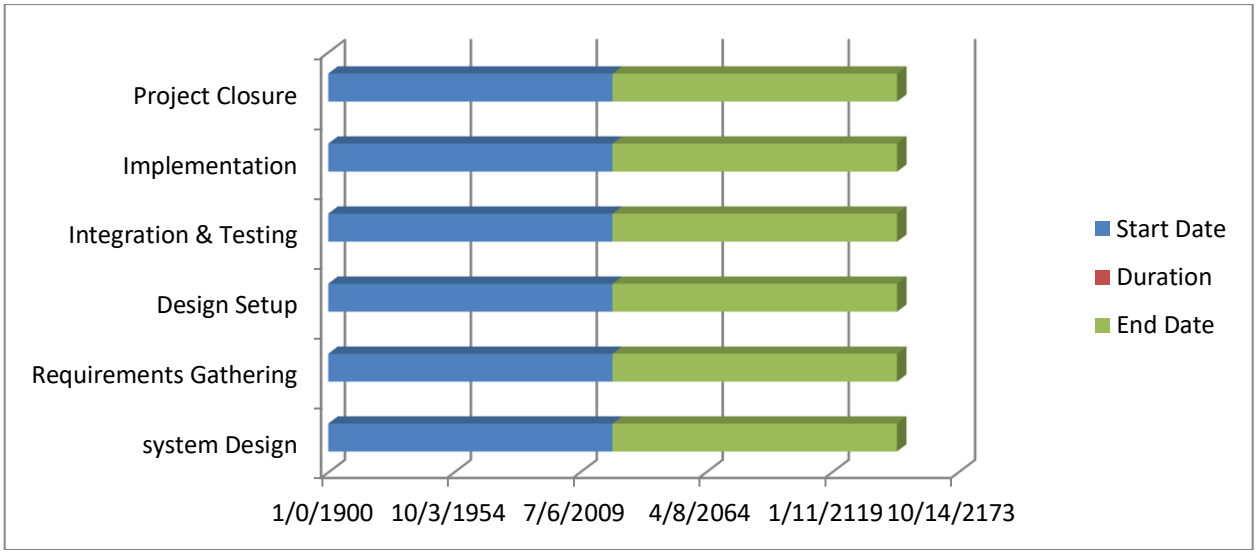
1.2.1 Technical Feasibility

The technical feasibility of the TMS is rooted in its robust architecture, which incorporates cutting-edge technologies such as C++ for efficient programming and SQL Server for seamless database management. The system's underlying infrastructure is capable of handling significant volumes of data while maintaining optimal performance. Rigorous testing and optimization will ensure that the system operates smoothly, providing users with a reliable and responsive experience.

1.2.2 Schedule Feasibility (MS Project Gantt Chart)

To ensure efficient project management and timely delivery, a detailed project schedule has been established using MS Project's Gantt Chart functionality. This schedule delineates key project phases, milestones, and dependencies. The Gantt Chart provides a visual representation of project progress, helping the project team allocate resources, manage tasks, and monitor overall project health.

Schedule Feasibility Gantt chart



1.3 Project Scope

We define the scope of our project, outlining its objectives, boundaries, and the specific functionalities it aims to deliver. This chapter provides a clear understanding of what the project intends to achieve.

The Traffic Management System will provide the following key features:

- Connected vehicles. ...
- Smart traffic signal control system. ...
- ANPR (automatic number plate recognition) ...
- Safety measures. ...
- Economic value. ...
- Environmental awareness.

1.4 Tools and Technology Used with Reasoning

The following tools and technologies are selected for the development of the Traffic Management System:

1.4.1 MS Project

We elaborate on the utilization of MS Project as a comprehensive project management tool. This section demonstrates how MS Project facilitates effective project planning, monitoring, and resource allocation.

1.4.2 VS Code

An insight into the choice of Visual Studio Code as the primary development environment is provided. We elucidate its features and functionalities that enhance coding efficiency, collaboration, and code quality.

1.5 Risk List

The risk list identifies potential risks and provides strategies to mitigate them effectively.

1.5.1 Time Risk

We identify potential time-related risks that may impact project schedules. Each risk is thoroughly analyzed, and contingency plans are outlined to mitigate any delays.

1.5.2 Compatibility Risk

The potential risks associated with system compatibility and integration are discussed. Strategies to ensure seamless integration and optimal system performance are explored.

Chapter 2: Requirement Engineering

2.1 Introduction

We delve into the essential process of requirement engineering, elucidating its significance in shaping the project's success. This chapter sets the stage for the detailed analysis of project requirements.

2.2 Functional Requirements

The system's core functions include

- Vehicle registration.
- Challan issuance.
- Challan payment.
- Violation record management.
- User authentication.
- User-friendly interfaces for administrators and users.
- Delete Vehicle
- Update Vehicle
- Search Vehicle

2.3 Non-Functional Requirements

The system's non-functional requirements encompass aspects such as.

- Performance **(2.3.1)**
- Efficiency **(2.3.2)**
- Security **(2.3.3)**
- User Experience. **(2.3.4)**

It must handle concurrent user interactions, process requests promptly, maintain data accuracy, and ensure secure data storage and transmission.

2.4 System Requirements

The TMS relies on a robust infrastructure that includes user devices, a central server, and network connectivity. User devices must meet the minimum hardware and software requirements to access the system, while the central server must possess sufficient processing power, storage capacity, and network bandwidth to support the data-intensive operations of the TMS. Specification of the operating system and software dependencies required for the TMS.
Compatibility with modern web browsers to ensure a seamless user experience.

2.5 End-User Requirements

This section focuses on understanding the needs and expectations of the various end-users who will interact with the Traffic Management System (TMS). By addressing the specific requirements of each user group, we ensure that the system provides a tailored and user-friendly experience.

- Access to a user-friendly dashboard showing vehicle violation records, pending challans, and payment options. **(2.5.1)**
- Ability to view violation details, including rule violations, dates, and fine amounts.
- Convenient payment portal for challan settlement, with options for online payment.
- Centralized administrative control panel for user management, system configuration, and reporting.

2.6 Data Requirements

This section outlines the data that the Traffic Management System (TMS) needs to capture, store, and manage to fulfill its functionality. Effective data management ensures accurate violation recording, reliable payment processing, and meaningful reporting.

- Capture and store user information, including vehicle owners, traffic officers, and administrators.
- Maintain user profiles, authentication credentials, and contact details.
- Store comprehensive violation records for each incident, including vehicle details, violation type, date, and location

Chapter 3: Implementation

3.1 Programming Languages and Frameworks

We provide insights into the selection of specific programming languages and frameworks, justifying their choice based on factors such as compatibility, efficiency, and developer expertise.

3.2 Coding Standards and Guidelines

The establishment of coding standards and guidelines is discussed, emphasizing the importance of consistent and high-quality coding practices. This section ensures that the developed codebase is maintainable, readable, and adheres to industry best practices.

3.3 User Interface Implementation

We detail the implementation of the user interface, highlighting the design principles, user interactions, and visual elements that contribute to an engaging and intuitive user experience.

3.4 Testing and Quality Assurance

A comprehensive overview of the testing strategies employed to verify the system's functionality, reliability, and adherence to requirements is presented. Quality assurance practices are outlined to ensure the delivery of a robust and error-free system.

3.5 Performance Testing

The execution of performance testing to assess the system's responsiveness, scalability, and resource utilization is discussed. This section ensures that the system meets performance expectations under various conditions.

Chapter 4: Deployment and Maintenance

4.1 Deployment Plan:

This section outlines the strategy for deploying the Traffic Management System (TMS) into the production environment. It provides a step-by-step plan to ensure a smooth transition from development to operational use.

Infrastructure Setup:

Detailed instructions for setting up the required hardware, software, and network infrastructure in the production environment. Deployment of the TMS components, including the web application, database, and any supporting services.

Data Migration:

Procedures for migrating data from the development or testing environment to the production database. Verification of data integrity and consistency after migration. User Training and Documentation:

Creation of user manuals, guides, and tutorials to assist end-users in navigating and utilizing the TMS effectively. Conducting training sessions to familiarize users, including vehicle owners, traffic officers, and administrators, with the system's features and functionality.

4.2 User Training and Documentation:

This subsection delves deeper into the process of user training and the creation of comprehensive documentation to ensure that all stakeholders can confidently interact with the Traffic Management System (TMS).

Training Programs:

Tailored training sessions for different user groups, including hands-on exercises to practice using the TMS. Online training resources, video tutorials, and FAQs to provide continuous learning opportunities.

4.3 Maintenance and Support Strategy:

This section outlines the approach to maintaining and supporting the TMS after deployment, ensuring its ongoing reliability, security, and performance.

Regular Maintenance:

Implementation of scheduled maintenance routines, including updates, patches, and system health checks. Proactive monitoring to identify and address potential issues before they impact users.

Technical Support:

Establishment of a helpdesk or support system to assist users with inquiries, technical issues, and troubleshooting.Timely response to user queries and the resolution of any system-related challenges.

Enhancements and Updates:

Process for introducing new features, functionality improvements, and system updates based on user feedback and evolving needs. Ensuring compatibility with emerging technologies and compliance with changing regulations.

Chapter 5: Conclusion

5.1 Summary of Achievements:

In this section, we summarize the key accomplishments and achievements of the Traffic Management System (TMS) project, highlighting its successful implementation, user adoption, and impact on traffic management efficiency.

System Functionality:

Recap of the TMS's core features and how they address traffic violation management needs.
Overview of the user-friendly interface and seamless user experience provided by the system.
User Adoption and Engagement:

Evaluation of user engagement and satisfaction levels through user feedback and usage metrics.
Recognition of how the TMS has streamlined processes and improved interaction between vehicle owners, traffic officers, and administrators.

5.2 Challenges Faced and Lessons Learned:

An honest reflection on the challenges encountered during the project's lifecycle and the valuable lessons gained from overcoming these obstacles.

Technical Challenges:

Discussion of any technical hurdles faced during development, deployment, or maintenance, along with the solutions devised to overcome them.
Operational Insights:

Reflection on how the TMS has impacted daily traffic management operations and any unexpected outcomes or insights gained.

5.3 Future Enhancements and Expansion:

A forward-looking section that explores potential avenues for further enhancing the Traffic Management System (TMS) and expanding its capabilities.

Feature Enhancements:

Exploration of additional features, such as real-time traffic data integration, predictive analytics, or integration with other smart city initiatives.
Scaling Possibilities:

Consideration of the system's scalability to accommodate increased user volume, data growth, and integration with future technologies.

Scaling Possibilities:

Consideration of the system's scalability to accommodate increased user volume, data growth, and integration with future technologies.

Chapter 6: Code and Output

final_oop.cpp

```cpp
// Header Files
#include <iostream>  //inout output Stream
#include <fstream>   // for File Handling
#include <sstream>   // convert data types between string to other
#include <string>    // for string Methods
#include <conio.h>   // For _getch() function
#include <windows.h> // For Sleep()  function
#include <iomanip>   // For setw() function
using namespace std;

// declaration of Functions
void login_sign_in_Menu();
void main_Menu();
void banner() // temprary function for main menu beautify
{
    system("cls");
    cout << "\n\n\n\n"; // Add multiple newlines for vertical centering

    cout << "\t\t\t\t\tWelcome to the Traffic Management System\n\n";

    cout << "\t\t\t\t\t-----------------------------------------\n\n";
}

void banner2() // temprary function for main menu beautify
{
    cout << "\n\n\n\n\n";
    cout << "\t\t\t\t\tWelcome to the sign-in and login page\n\n";
    cout << "\t\t\t\t\t-------------------------------------\n\n";
}

// Class User
class User
{
private:
    string username;
    string password;

public:
    // Public methods to access and modify the private attributes
    void setUsername(string uname) // set username
    {
        username = uname;
    }

    void setPassword(string pword) // set password
    {
        password = pword;
    }

    string getUsername() // get user name
    {
        return username;
```

```
    }

    string getPassword() // get password
    {
        return password;
    }

    string getPasswordInput() // Get password input and convert into staric *
    {
        string password;
        char ch;

        while ((ch = _getch()) != 13)
        { // 13 is the ASCII code for Enter key
            if (ch != '\r')
            { // Ignore carriage return character
                cout << '*';
                password.push_back(ch);
            }
        }

        // Clear any remaining characters in the input buffer
        while (_kbhit())
        {
            _getch();
        }
        return password;
    }

    bool isUsernameTaken(string username) // This Function perfomr the Username is exist or not in a file
    {
        ifstream file("users.txt"); // ifstream input file stream & file is the object of ifstream
        if (file.is_open())
        {
            string u, p;           // u for username and p for password
            while (file >> u >> p) // read data from file into two variables u and p
            {
                if (u == username) // condition compare and return true and false
                {
                    return true;
                }
            }
            file.close(); // file close
        }
        return false;
    }

    void updateUser() // Update Function: Update username and password
    {
        string newUsername;
        string newPassword;
        string existingname;
        cout << "\t\t\t\t\tEnter your Existing user name: "; // enter existing username to update account
        cin.ignore();
```

```
107         getline(cin, existingname);
108
109         bool userFound = false; // set value dalfe by default
110
111         ifstream inputFile("users.txt");     // iftream is input file stream
112         ofstream tempFile("temp_users.txt"); // ofstream is output file stream
113
114         if (inputFile.is_open() && tempFile.is_open()) // check condition if files is open or
    not
115         {
116             string line;
117             while (getline(inputFile, line)) // this loop read the line from input file using
    getline() function
118             {
119                 stringstream ss(line); // ss means stringstream break line into separate
    words, Each line is stored in the variable line
120                 string u, p;           // u for username & p for password
121                 ss >> u >> p;
122                 if (u == existingname) // check file for username is exist or not
123                 {
124                     cout << "\n\t\t\t\t\tEnter New username: ";
125                     getline(cin, newUsername);
126
127                     if (isUsernameTaken(newUsername)) // call isUsernameTaken() fun and check
    condition username is already exist or not
128                     {
129                         cout << "\n\t\t\t\t\tUsername already taken. Please choose another
    name.";
130                         cout << endl
131                              << "\t\t\t\t\twait for 2 sec ";
132                         Sleep(2000);
133                         system("cls");
134                         return;
135                     }
136                     cout << endl
137                          << "\t\t\t\t\tEnter new password: ";
138                     cin >> newPassword;
139                     tempFile << newUsername << " " << newPassword << endl; // store updated
    username and password in file
140                     userFound = true;
141                 }
142                 else
143                 {
144                     tempFile << line << endl; // If the username doesn't match the
    existingname, the code writes the entire line
145                 }
146             }
147
148             inputFile.close(); // close file
149             tempFile.close();  // close file
150
151             if (userFound) // the value of userFound variable is true
152             {
153                 if (remove("users.txt") != 0) // remove original file name using remove()
154                 {
155                     cout << "\n\t\t\t\t\tError deleting original file.\n";
156                 }
```

```
157                 if (rename("temp_users.txt", "users.txt") != 0) // rename the original file name using rename()
158                 {
159                     cout << "\n\t\t\t\t\tError renaming temporary file.\n";
160                 }
161 
162                 cout << "\n\t\t\t\t\tAccount updated successfully!";
163                 cout << endl
164                      << "\t\t\t\t\tWait for 2 seconds ";
165                 Sleep(2000);   // Sleep() function
166                 system("cls"); // clear consloe function()
167             }
168             else // if above condition false then run this code of block
169             {
170                 cout << "\n\t\t\t\t\tUser name not found. Please try again.";
171                 cout << endl
172                      << "\t\t\t\t\tWait for 2 seconds ";
173                 Sleep(2000);
174                 system("cls");
175             }
176         }
177         else // if above condition false then run this code of block
178         {
179             cout << "\n\t\t\t\t\tError updating the file.";
180             cout << endl
181                  << "\n\t\t\t\t\tWait for 2 seconds ";
182             Sleep(2000);
183             system("cls");
184         }
185     }
186 }; // End of class
187 
188 // Sign in class
189 class signin : public User // used Inheritance
190 {
191 public:
192     void signUp() // signup function to create account
193     {
194         User newUser; // Conatinership method
195         string username;
196         cout << "\t\t\t\tEnter username: ";
197         cin.ignore(); // ignore new line character
198         getline(cin, username);
199         cout << endl;
200 
201         if (isUsernameTaken(username)) // call isUsernameTaken() fun and check condition username is already exist or not
202         {
203             cout << "\n\t\t\t\t\tUsername already taken. Please choose another name.";
204             cout << endl
205                  << "\t\t\t\t\twait for 2 sec ";
206             Sleep(2000);
207             system("cls");
208             return;
209         }
210 
211         newUser.setUsername(username); // newuser is the object of clas User
```

```
212
213         if (username == "") // condition for the username and password is empty or not while creating account
214         {
215             cout << "\t\t\t\t\tThe username should not be empty : ";
216             cout << endl
217                  << "\t\t\t\t\twait for 2 sec ";
218             Sleep(2000);
219             system("cls");
220         }
221
222         else // if above condition false then run this code of block
223         {
224             cout << "\t\t\t\t\tEnter password: ";
225             newUser.setPassword(getPasswordInput()); // set password by calling getPasswordInput() fun
226
227             ofstream file("users.txt", ios::app); // ios::app means All output operations are performed at the end of the file
228             if (file.is_open())                  // condition for file is open or not
229             {
230                 file << newUser.getUsername() << " " << newUser.getPassword() << endl; // store username and pass in file
231                 file.close();                                                          // close file
232                 cout << endl
233                      << "\n\t\t\t\t\tAccount created successfully!";
234                 cout << endl
235                      << "\t\t\t\t\twait for 2 sec ";
236                 Sleep(2000);
237                 system("cls");
238             }
239             else // if above condition false then run this code of block
240             {
241                 cout << "\n\n\t\t\t\t\tError opening the file.";
242                 cout << endl
243                      << "\t\t\t\t\twait for 2 sec ";
244                 Sleep(2000);
245                 system("cls");
246             }
247         }
248     }
249
250     bool checkCredentials(string username, string password) // Bool function check Credentials and return True and False
251     {
252         ifstream file("users.txt");
253         if (file.is_open()) // condition for file is open or not
254         {
255             string u, p;           // u for username and p for password
256             while (file >> u >> p) // check condition & file is the object of ifstream
257             {
258                 if (u == username && p == password) // check condition and return true and false
259                 {
260                     return true;
261                 }
262             }
```

```
            file.close(); // close file
        }
        return false;
    }
}; // End of class

// Class Login
class login : public signin // used Inheritance
{
public:
    void logins() // login function for login
    {
        string username, password;
        cout << "\t\t\t\t\tEnter username: ";
        cin.ignore();
        getline(cin, username);
        cout << endl
             << "\t\t\t\t\tEnter password: ";
        password = getPasswordInput(); // set password by calling getPasswordInput() fun

        if (checkCredentials(username, password)) // check condition for login account
        {
            cout << "\n\n\t\t\t\t\tLogin successful!\n";
            cout << endl
                 << "\t\t\t\t\twait for 2 sec ";
            Sleep(2000);
            system("cls");
            main_Menu();
        }
        else // if the enters username and password are deos not exist in file then run this
code
        {
            cout << "\n\n\t\t\t\t\tInvalid username or password. Please try again.";
            cout << endl
                 << "\t\t\t\t\twait for 2 sec ";
            Sleep(2000);
            system("cls");
        }
    }

    void clearScreen() // clear Screen Method
    {
        system("cls"); // this function perfom task same as system("cls")
    }
}; // End of class

// Class Vehical
class Vehicle
{
private: // class vehicle private data member
    string registrationNumber;
    string owner;
    string vehicletype;
    string entryDate;
    bool challanStatus;
    string ruleBreak;
```

```
    string pbreak;
    int cpayment;
    int challanPayment;

public:
    // initialize data members by using parameterized   ructor
    Vehicle(string regNumber = "", string ownerName = "", string vehicleType = "")
    {
        registrationNumber = regNumber;
        owner = ownerName;
        vehicletype = vehicleType;
        challanStatus = false;
        ruleBreak = "";
        pbreak = "";
        cpayment = 0;
        challanPayment = 0;
        entryDate = "";
    }

    // Public Methods of class Vehicle
    string getRegistrationNumber() // get reg no
    {
        return registrationNumber;
    }

    string getOwnerName() // get owner name
    {
        return owner;
    }

    string getVehicleType() // get vehicle type
    {
        return vehicletype;
    }

    string getEntryDate() // get entry date of vehicle
    {
        return entryDate;
    }

    void setEntryDate(string entrydate) // set entry date details by parameters
    {
        entryDate = entrydate;
    }

    bool hasChallan() // has challan function used in some differnet places
    {
        return challanStatus;
    }

    void setChallanDetails(string rule, int payment) // set challan details by parameters
    {
        ruleBreak = rule;
        pbreak = rule;
        cpayment = payment;
        challanPayment = payment;
```

```
374         challanStatus = true;
375     }
376
377     void displayChallanDetails() // display challan details function and call it into other
    fun
378     {
379         cout << endl
380              << "\t\t\t\t\tChallan Date       : " << entryDate << "\n";
381         cout << "\t\t\t\t\tRule Break         : " << ruleBreak << "\n";
382         cout << "\t\t\t\t\tChallan Payment    : pkr " << challanPayment << "\n";
383     }
384
385     void cDeatils()
386     {
387         cout << "\n\t\t\t\t\tRule Break : " << pbreak <<endl;
388         cout << "\t\t\t\t\tChallan Payment : " << cpayment << "\n";
389     }
390
391     int getChallanPayment() // get challan payment
392     {
393         return challanPayment;
394     }
395
396     string getPaymentStatus() // get payment status
397     {
398         return challanStatus ? "Not Paid" : "Paid";
399     }
400
401     void setChallaned(bool challaned) // set challaned status by parameters type bhool true
    or false
402     {
403         challanStatus = challaned;
404     }
405
406     void setRegNumber(string newRegNumber) // Set reg No by parameters
407     {
408         registrationNumber = newRegNumber;
409     }
410
411     void setOwnerName(string newOwnerName) // set owner Name by parameters
412     {
413         owner = newOwnerName;
414     }
415
416     void setVehicleType(string newVehicleType) // set vehicle type by parameters
417     {
418         vehicletype = newVehicleType;
419     }
420 }; // End of class
421
422 // class TrafficManagmentSystem This is the most imp and main part of project
423 class TrafficManagementSystem : public Vehicle
424 {
425     // private Attributes of class TrafficManagmentSystem
426 private:
427     Vehicle vehicles[100]; // suppose maximum of 100 vehicles can be registered
428     int numVehicles = 0;
```

```
429     int totalChallanPayment;
430
431 public:
432     TrafficManagementSystem() //   Constructor
433     {
434         totalChallanPayment = 0;
435     }
436
437     // Register Vehicle Function
438     void registerVehicle(string regNumber, string ownerName, string vehicleType, string
    entryDate)
439     {
440         // Check if a vehicle with the same registration number and entry date already exists
441         bool vehicleExists = false;
442         for (int i = 0; i < numVehicles; ++i)
443         {
444             if (vehicles[i].getRegistrationNumber() == regNumber &&
    vehicles[i].getEntryDate() == entryDate)
445             {
446                 vehicleExists = true;
447                 break;
448             }
449         }
450
451         if (vehicleExists)
452         {
453             cout << "\n\t\t\t\t\t\tVehicle with the same registration number and entry date
    already exists.\n";
454         }
455         else
456         {
457             // If no matching vehicle is found, add the new vehicle
458             if (numVehicles < 100) // Ensure we don't exceed the maximum number of vehicles
459             {
460                 vehicles[numVehicles] = Vehicle(regNumber, ownerName, vehicleType);
461                 vehicles[numVehicles].setEntryDate(entryDate); // Set the entry date
462                 numVehicles++;
463                 cout << "\n\t\t\t\t\t\tVehicle registration successful.\n";
464             }
465             else
466             {
467                 cout << "\n\t\t\t\t\t\tMaximum number of vehicles reached. Cannot register more
    vehicles.\n";
468             }
469         }
470     }
471
472     // Issuance of challan Function
473     void issueChallan(string regNumber, string entryDate)
474     {
475         bool found = false;
476         for (int i = 0; i < numVehicles; ++i)
477         {
478             if (vehicles[i].getRegistrationNumber() == regNumber &&
    vehicles[i].getEntryDate() == entryDate)
479             {
480                 found = true;
```

```
481                 if (!vehicles[i].hasChallan())
482                 {
483                     string rule;
484                     int payment;
485                     cout << endl
486                          << "\t\t\t\t\tEnter Rule Break: ";
487                     getline(cin, rule);
488                     cout << endl
489                          << "\t\t\t\t\tEnter Challan Payment: ";
490                     cin >> payment;
491                     cin.ignore();
492 
493                     vehicles[i].setChallanDetails(rule, payment);
494                     totalChallanPayment += payment;
495                     cout << "\n\t\t\t\t\tChallan issued successfully.\n";
496                 }
497                 else
498                 {
499                     cout << "\n\t\t\t\t\tVehicle already has a challan.\n";
500                 }
501             }
502         }
503 
504         if (!found)
505         {
506             cout << "\n\t\t\t\t\tVehicle not found or entry date mismatch.\n";
507         }
508     }
509 
510     // Display Violation History of Vehicles
511     void displayViolationRecordsByDateAndType(string regNumber, string ownerName, string
    vehicleType, string entryDate)
512     {
513 
514         bool found = false;
515         for (int i = 0; i < numVehicles; ++i)
516         {
517             if (vehicles[i].getRegistrationNumber() == regNumber &&
    vehicles[i].getOwnerName() == ownerName && vehicles[i].getVehicleType() == vehicleType &&
    vehicles[i].getEntryDate() == entryDate)
518             {
519                 cout << "\n\t\t\t\t\tViolation Records for Given Data is\n\n";
520                 found = true;
521                 cout << "\t\t\t\t\tReg No: " << vehicles[i].getRegistrationNumber() << "\n";
522                 cout << "\t\t\t\t\tOwner : " << vehicles[i].getOwnerName() << "\n";
523                 cout << "\t\t\t\t\tType  : " << vehicles[i].getVehicleType() << "\n";
524                 vehicles[i].cDeatils();
525 
526                 if (vehicles[i].hasChallan())
527                 {
528                     vehicles[i].displayChallanDetails();
529                     cout << "\t\t\t\t\tPayment Status     : " <<
    vehicles[i].getPaymentStatus() << "\n";
530                 }
531             }
532         }
533         if (!found)
```

```
534         {
535             cout << "\n\t\t\t\t\t\tNo records found for the given date.....\n";
536         }
537     }
538     // Payment portal Function
539     void paymentPortal(string regNumber, string entryDate)
540     {
541         bool found = false;
542         for (int i = 0; i < numVehicles; ++i)
543         {
544             if (vehicles[i].getRegistrationNumber() == regNumber &&
    vehicles[i].getEntryDate() == entryDate)
545             {
546                 found = true;
547                 if (vehicles[i].hasChallan())
548                 {
549                     cout << "\n\t\t\t\t\tChallan Details:\n";
550                     vehicles[i].displayChallanDetails();
551                     cout << "\t\t\t\t\tPayment Status     : " <<
    vehicles[i].getPaymentStatus() << "\n";
552                     cout << endl
553                          << "\t\t\t\t\tEnter Payment Amount: pkr ";
554                     int paymentAmount;
555                     cin >> paymentAmount;
556                     if (paymentAmount >= vehicles[i].getChallanPayment())
557                     {
558                         int change = paymentAmount - vehicles[i].getChallanPayment();
559                         vehicles[i].setChallaned(false);
560                         cout << "\n\t\t\t\t\tPayment successful.\n";
561                         if (change > 0)
562                         {
563                             cout << endl
564                                  << "\t\t\t\t\tChange: pkr " << change << "\n";
565                         }
566                     }
567                     else
568                     {
569                         cout << endl
570                              << "\t\t\t\t\tInsufficient payment amount.\n";
571                     }
572                 }
573                 else
574                 {
575                     cout << "\n\t\t\t\t\tNo pending challan for this vehicle.\n";
576                 }
577             }
578         }
579
580         if (!found)
581         {
582             cout << "\n\t\t\t\t\tVehicle not found or entry date mismatch.\n";
583         }
584     }
585
586     // List of Violation Category function
587     void trafficViolationCategories()
588     {
```

```
589
590         // traffic violation categories here
591         cout << "\t\t\t\t\tTraffic Violation Categories And Rates:\n"
592              << endl;
593         cout << "\t\t\t\t\t1. Over Speeding         |        300  pkr\n";
594         cout << "\t\t\t\t\t2. Red Light Violation   |        250  pkr\n";
595         cout << "\t\t\t\t\t3. Illegal Parking       |        500  pkr\n";
596         cout << "\t\t\t\t\t4. No Helmit Wearing     |        200  pkr\n";
597         cout << "\t\t\t\t\t5. No Documents          |        1000 pkr\n";
598         cout << "\t\t\t\t\t6. Illegal Smuggling     |        1500 pkr\n";
599     }
600     // Display All vehicles functions
601     void displayAllVehicles()
602     {
603         cout << "\n\n";
604         cout << "\t\t\t\t\t+-----------------+------------------+--------------+--------------
+" << endl;
605         cout << "\t\t\t\t\t| Registration No | Owner Name       | Vehicle Type|  Entry Date
|" << endl;
606         cout << "\t\t\t\t\t+-----------------+------------------+--------------+--------------
+" << endl;
607
608         if (numVehicles <= 0)
609         {
610             cout << "\t\t\t\t\tNoo Vehicles Kindly add Vehicles first.......\n";
611         }
612         else
613         {
614             for (int i = 0; i < numVehicles; ++i)
615             {
616                 cout << "\t\t\t\t\t|" << setw(17) << left <<
vehicles[i].getRegistrationNumber()
617                      << "|" << setw(18) << left << vehicles[i].getOwnerName() << "|" <<
setw(13) << left << vehicles[i].getVehicleType();
618                 cout << "|" << setw(14) << left << vehicles[i].getEntryDate() << "|\n";
619                 cout << "\t\t\t\t\t+-----------------+------------------+--------------+----
--------+" << endl;
620             }
621         }
622     }
623     // Search Vehicles Function
624     void searchVehicle(string regNumber, string entryDate)
625     {
626         cout << "\n\n";
627         cout << "\t\t\t\t\t+------------------+------------------+--------------+--------------
+" << endl;
628         cout << "\t\t\t\t\t| Registration No | Owner Name       | Vehicle Type|  Entry Date
|" << endl;
629         cout << "\t\t\t\t\t+------------------+------------------+--------------+--------------
+" << endl;
630         bool found = false;
631         for (int i = 0; i < numVehicles; ++i)
632         {
633             if (vehicles[i].getRegistrationNumber() == regNumber &&
vehicles[i].getEntryDate() == entryDate)
634             {
635                 cout << "\t\t\t\t\t|" << setw(17) << left <<
vehicles[i].getRegistrationNumber()
```

```
636                     << "|" << setw(18) << left << vehicles[i].getOwnerName() << "|" << setw(13) << left << vehicles[i].getVehicleType();
637                 cout << "|" << setw(14) << left << vehicles[i].getEntryDate() << "|\n";
638                 cout << "\t\t\t\t\t+------------------+------------------+--------------+--------------+" << endl;
639                 found = true;
640                 break;
641             }
642         }
643         if (!found)
644         {
645             cout << "\t\t\t\t\tVehicle not found.\n";
646         }
647     }
648     // Delete Vehicles Function
649     void deleteVehicle(string regNumber, string entryDate)
650     {
651
652         for (int i = 0; i < numVehicles; ++i)
653         {
654             if (vehicles[i].getRegistrationNumber() == regNumber && vehicles[i].getEntryDate() == entryDate)
655             {
656                 vehicles[i] = vehicles[numVehicles - 1];
657                 numVehicles--;
658                 cout << "\n\t\t\t\t\tVehicle deleted successfully.\n";
659                 return;
660             }
661         }
662         cout << "\n\t\t\t\t\tVehicle not found.\n";
663     }
664     // Update Vehicles Function
665     void updateVehicle(string regNumber, string entryDate)
666     {
667         bool found = false;
668         for (int i = 0; i < numVehicles; ++i)
669         {
670             if (vehicles[i].getRegistrationNumber() == regNumber && vehicles[i].getEntryDate() == entryDate)
671             {
672                 found = true;
673                 string newRegNumber, newOwnerName, newVehicleType;
674
675                 cout << endl
676                      << "\t\t\t\t\tEnter new Registration No: ";
677                 getline(cin, newRegNumber);
678
679                 cout << endl
680                      << "\t\t\t\t\tEnter new Owner Name: ";
681                 getline(cin, newOwnerName);
682
683                 cout << endl
684                      << "\t\t\t\t\tEnter new Vehicle Type: ";
685                 getline(cin, newVehicleType);
686
687                 vehicles[i].setRegNumber(newRegNumber);
688                 vehicles[i].setOwnerName(newOwnerName);
```

```
689                 vehicles[i].setVehicleType(newVehicleType);
690 
691                 cout << "\n\t\t\t\t\t\tVehicle information updated successfully.\n";
692             }
693         }
694 
695         if (!found)
696         {
697             cout << "\n\t\t\t\t\t\tVehicle not found or entry date mismatch.\n";
698         }
699     }
700 
701     // Show total challaned Payment History
702     int getTotalChallanPayment()
703     {
704 
705         return totalChallanPayment;
706     }
707 }; // End of class
708 
709 // Login_sign_on_Menu() fun
710 void login_sign_in_Menu()
711 {
712     login l1;         // object of class login
713     l1.clearScreen(); // for clear consloe
714     int choice;
715     do
716     {
717         cout << "\n\n\n\n\n\n"; // Add multiple newlines for vertical centering
718 
719         cout << "\t\t\t\t\tWelcome to the sign-in and login page\n\n";
720 
721         cout << "\t\t\t\t\t————————————————————————————————————\n\n";
722         cout << "\t\t\t\t\t1. Sign up\n";
723         cout << "\t\t\t\t\t2. Log in\n";
724         cout << "\t\t\t\t\t3. Update Info\n";
725         cout << "\t\t\t\t\t4. Exit\n"
726              << endl;
727         cout << "\t\t\t\t\tEnter choice: ";
728         cin >> choice;
729 
730         switch (choice)
731         {
732         case 1:
733             l1.clearScreen();
734             banner2();
735             cout << "\t\t\t\t\tKindly Sign up your self\n\n";
736             l1.signUp();
737             break;
738         case 2:
739             l1.clearScreen();
740             banner2();
741             cout << "\t\t\t\t\tNow Login your self\n\n";
742             l1.logins();
743             break;
744         case 3:
```

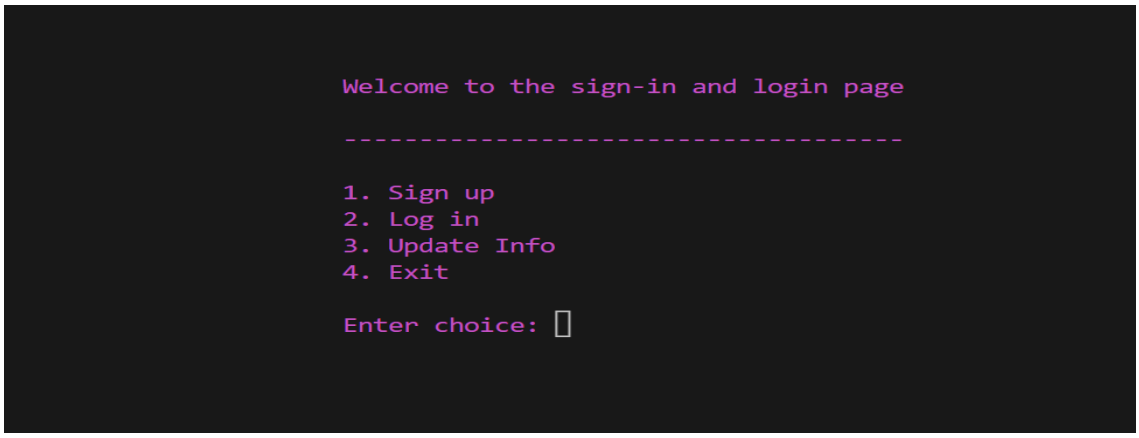
```
            l1.clearScreen();
            banner2();
            cout << "\t\t\t\t\tNow Update your Login Info\n\n";
            l1.updateUser();
            break;
        case 4:
            l1.clearScreen();
            cout << "Exiting th system...\n";
            break;
        default:
            cout << "\n\t\t\t\t\t\tInvalid choice. Please try again.\n";
            cout << "\t\t\t\t\twait for 2 sec ";
            Sleep(2000);
            system("cls");
        }
    } while (choice != 4);
} // End login_sign_in_Menu() fun

// Main Menu() Fun
void main_Menu()
{
    TrafficManagementSystem tms; // tms is the object of class TrafficManagmentSystem

    while (true) // Infinite Loop
    {
        banner(); // Call Banner function for display Headings

        cout << "\t\t\t\t\tMain Menu:\n"
             << endl;
        cout << "\t\t\t\t\t1.  Vehicle Registration\n";
        cout << "\t\t\t\t\t2.  Challan Issuance\n";
        cout << "\t\t\t\t\t3.  Violation Records\n";
        cout << "\t\t\t\t\t4.  Payment Portal\n";
        cout << "\t\t\t\t\t5.  Traffic Violation Categories\n";
        cout << "\t\t\t\t\t6.  Show all vehicles\n";
        cout << "\t\t\t\t\t7.  Search Vehicle\n";
        cout << "\t\t\t\t\t8.  Delete Vehicle\n";
        cout << "\t\t\t\t\t9.  Update Vehicle\n";
        cout << "\t\t\t\t\t10. Total Challan Payment\n";
        cout << "\t\t\t\t\t11. Logout\n"
             << endl;

        int choice;
        cout << "\t\t\t\t\tEnter Choice: ";
        cin >> choice;

        if (choice == 1)
        {
            banner();
            string regNumber, ownerName, vehicleType, entryDate;
            cout << "\t\t\t\t\tVehicle Registration Page: \n\n";
            cout << "\t\t\t\t\tEnter Vehicle Registration Number: ";
            cin.ignore();
            cin >> regNumber;
            cout << endl
                 << "\t\t\t\t\tEnter Owner Name: ";
```

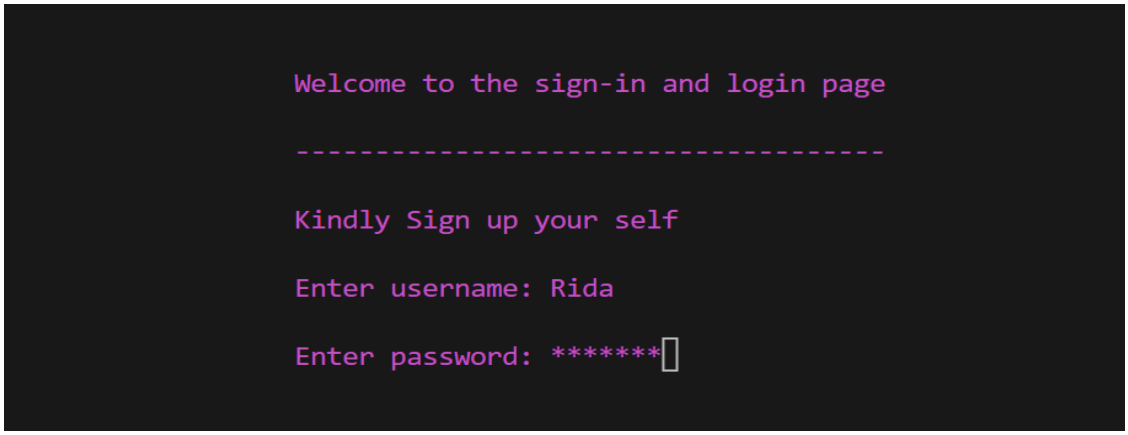
```
            cin.ignore();
            getline(cin, ownerName);
            cout << endl
                 << "\t\t\t\t\tEnter Vehicle Type: ";
            getline(cin, vehicleType);
            cout << endl
                 << "\t\t\t\t\tEnter Date (dd/mm/yyyy): ";
            getline(cin, entryDate);
            tms.registerVehicle(regNumber, ownerName, vehicleType, entryDate);
        }
        else if (choice == 2)
        {
            banner();
            cout << "\t\t\t\t\tVehicle Challan Issuance Page: \n\n";
            string regNumber, entryDate;
            cout << "\t\t\t\t\tEnter Vehicle Registration Number: ";
            cin >> regNumber;
            cout << "\n\t\t\t\t\tEnter Challan Issue Date (yyyy-mm-dd): ";
            cin.ignore();
            getline(cin, entryDate);
            tms.issueChallan(regNumber, entryDate);
        }
        else if (choice == 3)
        {
            banner();
            cout << "\t\t\t\t\tViolation Record Page: \n\n";
            string regNumber, ownerName, vehicleType, entryDate;
            cout << "\t\t\t\t\tEnter Registration Number: ";
            cin.ignore();
            getline(cin, regNumber);
            cout << endl
                 << "\t\t\t\t\tEnter Owner Name: ";
            getline(cin, ownerName);
            cout << endl
                 << "\t\t\t\t\tEnter Vehicle Type: ";
            getline(cin, vehicleType);
            cout << endl
                 << "\t\t\t\t\tEnter Date (dd/mm/yyyy): ";
            getline(cin, entryDate);
            tms.displayViolationRecordsByDateAndType(regNumber, ownerName, vehicleType,
entryDate);
        }
        else if (choice == 4)
        {
            banner();
            cout << "\t\t\t\t\tPayment Portal Page: \n\n";
            string regNumber, entryDate;
            cout << "\t\t\t\t\tEnter Vehicle Registration Number: ";
            cin >> regNumber;
            cout << "\n\t\t\t\t\tEnter Challan Issue Date (yyyy-mm-dd): ";
            cin.ignore();
            getline(cin, entryDate);
            tms.paymentPortal(regNumber, entryDate);
        }
        else if (choice == 5)
        {
```

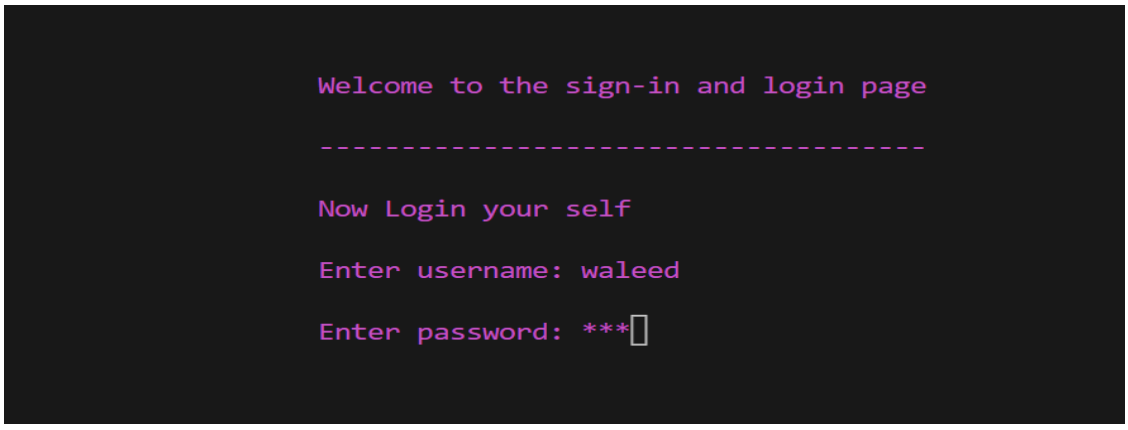
```
            banner();
            cout << "\t\t\t\t\tTraffic Violation Categories Page: \n\n";
            tms.trafficViolationCategories();
        }
        else if (choice == 6)
        {
            banner();
            tms.displayAllVehicles();
        }
        else if (choice == 7)
        {
            banner();
            cout << "\t\t\t\t\tVehicle Search Page: \n\n";
            string regNumber, entryDate;
            cout << "\t\t\t\t\tEnter Vehicle Registration Number: ";
            cin >> regNumber;
            cout << endl
                 << "\t\t\t\t\tEnter Entry Date (dd/mm/yyyy): ";
            cin.ignore();
            getline(cin, entryDate);
            tms.searchVehicle(regNumber, entryDate);
        }
        else if (choice == 8)
        {
            banner();
            cout << "\t\t\t\t\tVehicle Deletation Page: \n\n";
            string regNumber, entryDate;
            cout << "\t\t\t\t\tEnter Vehicle Registration Number: ";
            cin >> regNumber;
            cout << endl
                 << "\t\t\t\t\tEnter Entry Date (dd/mm/yyyy): ";
            cin.ignore();
            getline(cin, entryDate);
            tms.deleteVehicle(regNumber, entryDate);
        }

        else if (choice == 9)
        {
            banner();
            cout << "\t\t\t\t\tVehicle Update Page: \n\n";
            string regNumber, entryDate;
            cout << "\t\t\t\t\tEnter Vehicle Registration Number: ";
            cin >> regNumber;
            cout << "\n\t\t\t\t\tEnter Challan Issue Date (yyyy-mm-dd): ";
            cin.ignore();
            getline(cin, entryDate);
            tms.updateVehicle(regNumber, entryDate);
        }

        else if (choice == 10)
        {
            banner();
            cout << "\t\t\t\t\tTotal Challan Payment Page: \n\n";
            cout << "\t\t\t\t\tTotal Challan Payment: pkr " << tms.getTotalChallanPayment()
<< "\n";
        }
```

```cpp
        else if (choice == 11)
        {
            system("cls");
            cout << "\n\n\n\n\n\n";
            cout << "\t\t\t\t\t\tLogged out successfully.\n"
                 << endl;
            cout << "\t\t\t\t\t\twait for 2 sec ";
            Sleep(2000);
            system("cls");
            break;
        }
        else
        {
            cout << endl
                 << "\t\t\t\t\tInvalid choice. Please try again.\n";
        }
        cout << "\n\t\t\t\t\tPress Enter to continue...";
        getch();
    }
} // End of Main Menu() Fun

// Main Function Entery point of the project
int main()
{
    system("Color 05"); // change console color
    login_sign_in_Menu();

    return 0;
}
// End of project
```

Outputs:


Welcome to the sign-in and login page
1. Sign up
2. Log in
3. Update Info
4. Exit
Enter choice:


Welcome to the sign-in and login page
Kindly Sign up your self
Enter username: Rida
Enter password: *******


Welcome to the sign-in and login page
Now Login your self
Enter username: waleed
Enter password: ***

```
Welcome to the sign-in and login page

-------------------------------------

Now Update your Login Info

Enter your Existing user name: waleed

Enter New username: khattak

Enter new password: 12345
```

```
Welcome to the Traffic Management System

----------------------------------------

Main Menu:

1.  Vehicle Registration
2.  Challan Issuance
3.  Violation Records
4.  Payment Portal
5.  Traffic Violation Categories
6.  Show all vehicles
7.  Search Vehicle
8.  Delete Vehicle
9.  Update Vehicle
10. Total Challan Payment
11. Logout

Enter Choice:
```

```
Welcome to the Traffic Management System

----------------------------------------

Vehicle Registration Page:

Enter Vehicle Registration Number: 12345

Enter Owner Name: Zain

Enter Vehicle Type: Car

Enter Date (dd/mm/yyyy): 12 12 2000
```

```
----------------------------------------

Violation Record Page:

Enter Registration Number: 12345

Enter Owner Name: Zain

Enter Vehicle Type: Car

Enter Date (dd/mm/yyyy): 12 12 2000

Violation Records for Given Data is

Reg No: 12345
Owner : Zain
Type  : Car

Rule Break : Speeding
Challan Payment : 300

Challan Date       : 12 12 2000
Rule Break         : Speeding
Challan Payment    : pkr 300
Payment Status     : Not Paid

Press Enter to continue...
```

Ln 458, Col 92 Spaces: 4

```
Welcome to the Traffic Management System

----------------------------------------

Traffic Violation Categories Page:

Traffic Violation Categories And Rates:

1. Over Speeding          |        300  pkr
2. Red Light Violation    |        250  pkr
3. Illegal Parking        |        500  pkr
4. No Helmit Wearing      |        200  pkr
5. No Documents           |        1000 pkr
6. Illegal Smuggling      |        1500 pkr

Press Enter to continue...
```

```
Welcome to the Traffic Management System

---------------------------------------



+-----------------+-----------------+--------------+--------------+
| Registration No | Owner Name      | Vehicle Type|  Entry Date  |
+-----------------+-----------------+--------------+--------------+
|12345            |Zain             |Car           |12 12 2000    |
+-----------------+-----------------+--------------+--------------+
|123456           |Waleed           |Bik3          |02 11 1998    |
+-----------------+-----------------+--------------+--------------+

Press Enter to continue...
```

```
Welcome to the Traffic Management System

----------------------------------------

Payment Portal Page:

Enter Vehicle Registration Number: 12345

Enter Challan Issue Date (yyyy-mm-dd): 12 12 2000

Challan Details:

Challan Date        : 12 12 2000
Rule Break          : Speeding
Challan Payment     : pkr 300
Payment Status      : Not Paid

Enter Payment Amount: pkr 300

Payment successful.

Press Enter to continue...
```

```
Welcome to the Traffic Management System

----------------------------------------

Vehicle Deletation Page:

Enter Vehicle Registration Number: 12345

Enter Entry Date (dd/mm/yyyy): 12 12 2000

Vehicle deleted successfully.

Press Enter to continue...
```

```
Welcome to the Traffic Management System

-------------------------------------

Vehicle Update Page:

Enter Vehicle Registration Number: 123456

Enter Challan Issue Date (yyyy-mm-dd): 02 11 1998

Enter new Registration No: 0010

Enter new Owner Name: Khattak

Enter new Vehicle Type: car

Vehicle information updated successfully.

Press Enter to continue...
```

```
Exiting th system...
PS C:\Users\SUPER BRANDS COMPUTE\Documents\final>
```