**Railway Management System**



**Session 2023 – 2027**

**Submitted by:**

Mobeen Butt 2023-CS-28

**Supervised by:**

Prof. Awais Hassan

**Course:**

CSC-102 Programming Fundamentals

**Department of Computer Science**

**University of Engineering and Technology Lahore**

**Contents**

[1.1 Introduction 4](#_Toc153844903)

[1.2 Project Scope 4](#_Toc153844904)

[1.3 Users of Application 4](#_Toc153844905)

[1.4 Admin 5](#_Toc153844906)

[1.4.1 Login 5](#_Toc153844907)

[1.4.2 Sign Up 5](#_Toc153844908)

[1.4.3 Sign In 5](#_Toc153844909)

[1.5 Note for Login: 5](#_Toc153844910)

[1.5.1 Add New Train 5](#_Toc153844911)

[1.5.2 View Added Train 5](#_Toc153844912)

[1.5.3 Remove Added Trains 5](#_Toc153844913)

[1.5.4 Available Trains 5](#_Toc153844914)

[1.5.5 Information 6](#_Toc153844915)

[1.5.6 Ticket Information 6](#_Toc153844916)

[1.5.7 Helpline 6](#_Toc153844917)

[1.5.8 E-Ticket 6](#_Toc153844918)

[1.5.9 Logout 6](#_Toc153844919)

[1.6 User 6](#_Toc153844920)

[1.6.1 Login 7](#_Toc153844921)

[1.6.2 Sign Up 7](#_Toc153844922)

[1.6.3 Sign In 7](#_Toc153844923)

[1.7 Note for Login 7](#_Toc153844924)

[1.7.1 Home 7](#_Toc153844925)

[1.7.2 Train Timings 7](#_Toc153844926)

[1.7.3 Rates 7](#_Toc153844927)

[1.7.4 Information 7](#_Toc153844928)

[1.7.5 E-Booking 7](#_Toc153844929)

[1.7.6 Feedback and Ratings 7](#_Toc153844930)

[1.7.7 Ticket Information 8](#_Toc153844931)

[1.7.8 Helpline 8](#_Toc153844932)

[1.7.9 Logout 8](#_Toc153844933)

[1.8 Wireframes 8](#_Toc153844934)

[1.9 Functions Prototypes 12](#_Toc153844935)

[2.0 Complete Code 14](#_Toc153844936)

[2.1 Functions Working Flow 47](#_Toc153844937)

[2.2 Data Structure 47](#_Toc153844938)

[2.3 Weakness in the Application 50](#_Toc153844939)

[2.3.1 Account Security 50](#_Toc153844940)

[2.3.2 Single Responsibility Function 50](#_Toc153844941)

[2.3.3 Limited Access 50](#_Toc153844942)

[2.3.4 Not Access of Berth 50](#_Toc153844943)

[2.3.5 Class of Passenger 50](#_Toc153844944)

[2.4 Future Directions 50](#_Toc153844945)

# 1.1 Introduction

I am pleased to submit this proposal for the development of a Railway Management System (RMS) application. The proposed application aims to modernize and streamline Railway operations, improving efficiency, safety, and customer experience. This application will encompass various modules, including ticket booking, train scheduling, passenger management, feedback, ratings and more, to meet the diverse needs of a comprehensive Railway Management System.

# 1.2 Project Scope

* Passenger Information and Service
* Reporting and Analytics
* Customer Support and Feedback
* Mobile Application
* Online Ticket Booking
* Train Scheduling and Management

# 1.3 Users of Application

|  |  |  |
| --- | --- | --- |
| **Admin** | Add Train | Add new train. |
| Remove Train | Remove added train. |
| View Added Train | View added or remove trains. |
| Helpline | Get 24/7 help for any query. |
| Available Trains | View the available trains. |
| Information | Information about the application. |
| E-Ticket | Buy online tickets. |
| Ticket Information | Get tickets information. |
| Exit | Exit completely from application. |
| Log out | Log out from the Application. |
| **User** | Home | View the Services of our Train. |
| Train Timings | View Train timings. |
| Rates | View Rates or Fares of Trains. |
| Information | Information about the application. |
| E-Booking | Online Booking of Trains Ticket. |
| Feedback | Give Feedback to our application. |
| Ratings | Give Ratings to our application and services. |
| Ticket Information | Get tickets information. |
| Helpline | Get 24/7 help for any query. |
| Log Out | Log out from the application. |

# 1.4 Admin

Admin as administrator can add, remove, view trains also buy e-ticket.

## 1.4.1 Login

I add the login functionality. It improve the system by adding a strong login feature. This will allow users to sign in or sign up securely, giving them access to personalized features and information.

## 1.4.2 Sign Up

I add the Sign up function and introduce a user-friendly sign-up process where users can effortlessly create accounts. They will be prompted to provide essential information and preferences, ensuring a personalized experience according to their needs.

## 1.4.3 Sign In

I add the Sign In function and implement a secure sign-in process to authenticate users and grant access to their accounts. This feature enhances the security of user data.

# 1.5 Note for Login:

Login function is for Admin and Passenger. A user may be Admin or Passenger so, for Admin user must enter **Admin** and Passenger must enter **User** in the role choice.

## 1.5.1 Add New Train

Facilitate the addition of new trains to the system through a user-friendly and straightforward process. Administrators can effortlessly input essential train details such as the train name, destination, and departure time. The system ensures a smooth handling of this information, guaranteeing a hassle-free experience when integrating new trains into the Railway Management System.

## 1.5.2 View Added Train

Facilitate administrators in easily viewing added trains through a user-friendly process. This feature provides access to a comprehensive list of all system trains, highlighting essential details like train name, destination, and departure time. The interface ensures smooth monitoring and management, contributing to the system's overall efficiency by offering a clear overview of available trains. This enhancement streamlines train-related tasks for administrators.

## 1.5.3 Remove Added Trains

Admin can also remove added trains by Entering the Name of the train he/she wants to remove according to desired needs.

## 1.5.4 Available Trains

Here you check Available trains and track the train. For this purpose, we have trains name as follows:

* Pak-Train
* Yaadgar Express
* Jaffer Express
* Green Train

You can also check train for specific locations like:

* Lahore
* Karachi
* Peshawar
* Multan

## 1.5.5 Information

This function provides users with information about the railway system, historical data, and upcoming projects.

## 1.5.6 Ticket Information

It displays fare breakdowns for different routes and locations. User can get information about trains and ticket rates according to their desired journey.

## 1.5.7 Helpline

A 24/7 helpline assists users with queries, providing auto responses and connecting users with live agents for more complex issues. Users can get any kind of help about desired trains and any issue related to Railway Management System.

## 1.5.8 E-Ticket

Users can book e-tickets, selecting seats and providing preferences. The system calculates ticket prices based on factors like destination. Also display the information of passenger according to booked ticket.

## 1.5.9 Logout

User also logout from their accounts either Admin or Passenger (User) and change, create and add more accounts according to desired needs.

# 1.6 User

User may be Passenger, who get many facilities like creating account, check Train Timings, give Feedback, give Ratings, gets Information and also buy number of online tickets according to their needs.

# 1.6.1 Login

I add the login functionality. It improve the system by adding a strong login feature. This will allow users to sign in or sign up securely, giving them access to personalized features and information.

## 1.6.2 Sign Up

I add the Sign up function and introduce a user-friendly sign-up process where users can effortlessly create accounts. They will be prompted to provide essential information and preferences, ensuring a personalized experience according to their needs.

## 1.6.3 Sign In

I add the Sign In function and implement a secure sign-in process to authenticate users and grant access to their accounts. This feature enhances the security of user data.

# 1.7 Note for Login

Login function is for Admin and Passenger. A user may be Admin or Passenger so, for Admin user must enter **Admin** and Passenger must enter **User** in the role choice.

## 1.7.1 Home

Here Passenger, can check our services which we provide to our respected passengers. We provide best facility to our users.

## 1.7.2 Train Timings

Here our Passengers, check the trains timings by entering the train names like as follows:

* Pak Train
* Yaad Gar Express
* Jaffer Express
* Green Train

## 1.7.3 Rates

Passengers can check the fares and rates of trains according to Train name, location and destination. We provide reasonable fares to our users, that’s our priority.

## 1.7.4 Information

This function provides users with information about the railway system, historical data, and upcoming projects.

## 1.7.5 E-Booking

Passengers can book e-tickets, selecting seats and providing preferences. The system calculates ticket prices based on factors like destination. Also display the information of passenger according to booked ticket.

## 1.7.6 Feedback and Ratings

Passengers can give feedback to our facility and Railway Management System. It includes a ratings feature, allowing users to express their satisfaction levels. So that we can improve our facilities.

## 1.7.7 Ticket Information

Here Passenger add information of its ticket and verify ticket is available or not. Passengers provide their name, cnic, location and some more items for verification.

## 1.7.8 Helpline

A 24/7 helpline assists users with queries, providing auto responses and connecting users with live agents for more complex issues. Users can get any kind of help about desired trains and any issue related to Railway Management System.

## 1.7.9 Logout

User also logout from their accounts either Admin or Passenger (User) and change, create and add more accounts according to desired needs.

# 1.8 Wireframes

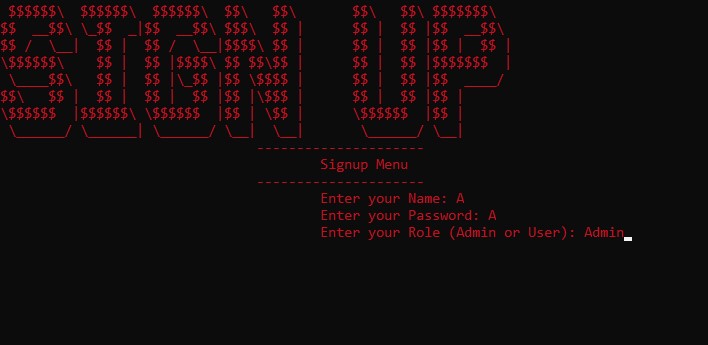
My login screen looks like that and its command line interface is displayed below:

****

Figure 1: Login Screen

****

**Figure 2: Sign In Screen**

****

**Figure 4: Sign Up Screen**



Figure 5: Admin Menu Screen

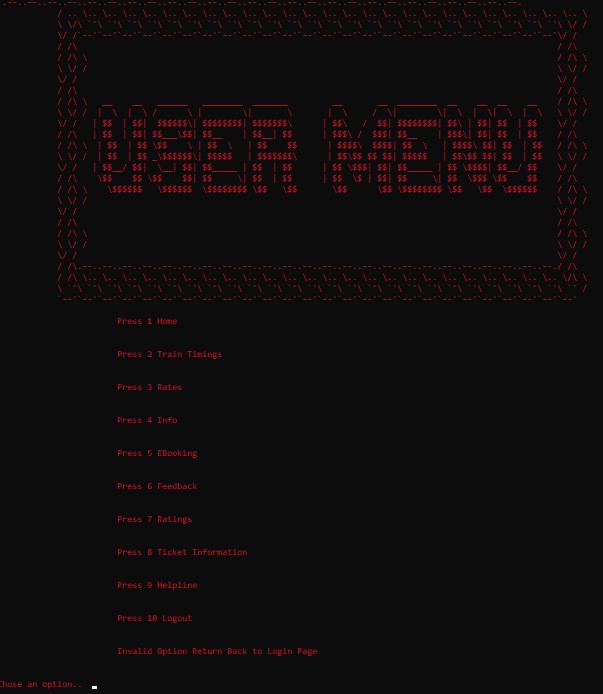
****

Figure 6: User Menu Screen



**Figure 7: Home Screen**

# 1.9 Functions Prototypes

My functions prototypes are displayed below:

1. #include <fstream>
2. #include <iostream>
3. #include <conio.h>
4. #include <windows.h>
5. #include <iomanip>
6. #include <string>
7. // My User Menu
8. void User();
9. // SERVICES
10. void services();
11. // Rates
12. void rate();
13. //Menu
14. void dashboard();
15. /// Header
16. void header();
17. //E-Ticket
18. void eticket();
19. //Feedbac);
20. void feedback();
21. /// User Rating
22. void rating();
23. //Information
24. void info();
25. //Helpline
26. void helpline();
27. //SUB MENU
28. void Menu(string submenu);
29. void subMenu(string submenu);
30. //ADMIN FUNCT
31. ///ADD information
32. void addTicketInformation();
33. /// Displaying Menu
34. void displayMenu();
35. /// View Added Trains By Admin
36. void viewTrains(const string names[], const string destinations[], const int departureTimes[], int numTrains);
37. // Display all user Information
38. void displayInformation(string names[], string cnics[], string locations[], int nums[], int totalAmounts[], int numTickets);
39. // Manage all Trains
40. void manageTrains(string names[], string destinations[], int departureTimes[], int &numTrains, const int MAX\_TRAINS);
41. // Remove Added Trains
42. void removeTrain(string names[], string destinations[], int departureTimes[], int &numTrains);
43. // Train Added by Admin
44. void addTrain(string names[], string destinations[], int departureTimes[], int &numTrains, const int MAX\_TRAINS);
45. /// File Handling
46. // Load Login Information
47. void loadUserRecords(string users[], string passwords[], string roles[], int &numUsers);
48. // Save Login Information
49. void saveUserRecord(string name, string password, string role);
50. // CSV fom Data
51. string getField(string record, int field);
52. string getFields(const string &record, int index);
53. //New File For more Data
54. void loadtrainFile(string names[], string destinations[], int departureTimes[], int &numTrains);
55. void trainSaveFile( string names[], string destinations[], int departureTimes[], int numTrains);
56. void removeTrainFromFile(const string &trainName);
57. //Login
58. int loginMenu();
59. //Sign in Function
60. string Signin(string name, string password, string users[], string passwords[], string roles[], int count);
61. //Sign Up function
62. bool Signup(string name, string password, string role, string users[], string passwords[], string roles[], int &usersCount, int userArrSize);
63. /// HEADERS
64. // LOGIN HEADER
65. void loginHeader();
66. // SIGN UP HEADER
67. void signupHeader();
68. /// SIGN IN HEADER
69. void signinHeader();
70. // ADMIN Header
71. void adminHeader();
72. /// USER Header
73. void userHeader();
74. /// Home HEADER
75. void home();
76. // Other Functions For Clear the Screen And Exit the program
77. void clearScreen();
78. void Exit();
79. //file handling for feedback
80. void saveFeedbackToFile(const string& res);

# 2.0 Complete Code

#include <fstream>

#include <iostream>

#include <conio.h>

#include <windows.h>

#include <iomanip>

#include <string>

using namespace std;

*// My User Menu*

void User();

*// SERVICES*

void services();

*//  Rates*

void rate();

*//Menu*

void dashboard();

*/// Header*

void header();

*//E-Ticket*

void eticket();

*//Feedback;*

void feedback();

*/// User Rating*

void rating();

*//Information*

void info();

*//Helpline*

void helpline();

*//SUB MENU*

void Menu(string submenu);

void subMenu(string submenu);

*//ADMIN FUNCT*

*///ADD information*

void addTicketInformation();

*/// Displaying Menu*

void displayMenu();

*/// View Added Trains By Admin*

void viewTrains(const string names[], const string destinations[], const int departureTimes[], int numTrains);

*// Display all user Information*

void displayInformation(string names[], string cnics[], string locations[], int nums[], int totalAmounts[], int numTickets);

*// Manage all Trains*

void manageTrains(string names[], string destinations[], int departureTimes[], int &numTrains, const int MAX\_TRAINS);

*// Remove Added Trains*

void removeTrain(string names[], string destinations[], int departureTimes[], int &numTrains);

*// Train Added by Admin*

void addTrain(string names[], string destinations[], int departureTimes[], int &numTrains, const int MAX\_TRAINS);

*/// File Handling*

*// Load Login Information*

void loadUserRecords(string users[], string passwords[], string roles[], int &numUsers);

*// Save Login Information*

void saveUserRecord(string name, string password, string role);

*// CSV fom Data*

string getField(string record, int field);

string getFields(const string &record, int index);

*//New File For more Data*

void loadtrainFile(string names[], string destinations[], int departureTimes[], int &numTrains);

void trainSaveFile( string names[],  string destinations[],  int departureTimes[], int numTrains);

void removeTrainFromFile(const string &trainName);

*//Login*

int loginMenu();

*//Sign in Function*

string Signin(string name, string password, string users[], string passwords[], string roles[], int count);

*//Sign Up function*

bool Signup(string name, string password, string role, string users[], string passwords[], string roles[], int &usersCount, int userArrSize);

*/// HEADERS*

*// LOGIN HEADER*

void loginHeader();

*// SIGN UP HEADER*

void signupHeader();

*/// SIGN IN HEADER*

void signinHeader();

*// ADMIN Header*

void adminHeader();

*/// USER Header*

void userHeader();

*/// Home HEADER*

void home();

*// Other Functions For Clear the Screen And Exit the program*

void clearScreen();

void Exit();

*//file handling for feedback*

void saveFeedbackToFile(const string& res);

int main()

{

    string trainName;

*//////array size for no more than 100 useres*

    const int userArrSize = 100;

    string users[userArrSize];

    string passwords[userArrSize];

    string roles[userArrSize];

     int count = 0;*////intialize count to 0*

    loadUserRecords(users, passwords, roles, count);

    int option = 0;

    while (option != 3)*//until option not greter than 3*

    {

        loginHeader();

        Menu("Login");

        option = loginMenu();

        if (option == 1)

        {

            system("cls");

            string name;

            string password;

            string role;

          signinHeader();

            Menu("Signin");

            cin.clear();*//clear input buffer*

            cin.sync();

            cout << "Enter your Name: ";

            getline(cin , name);

            cout << "Enter your Password: ";

            getline(cin ,password);

            role = Signin(name, password, users, passwords, roles, count);

            system("cls");

            if (role == "Admin")

            {

                clearScreen();

                const int MAX\_TRAINS = 10; *// Maximum number of trains*

                header();

                string names[MAX\_TRAINS];

                string destinations[MAX\_TRAINS];

                int departureTimes[MAX\_TRAINS];

                int numTrains = 0;

                int choice = 0;

                while (choice != 9)

                {*//////admin header*

                    adminHeader();

                    cout << "\n\n\t\t\tPress 1 Add New Train" << endl;

                    cout << "\n\n\t\t\tPress 2 View added Trains " << endl;

                    cout << "\n\n\t\t\tPress 3 Available Trains" << endl;

                    cout << "\n\n\t\t\tPress 4 Info" << endl;

                    cout << "\n\n\t\t\tPress 5 Ticket Information" << endl;

                    cout << "\n\n\t\t\tPress 6 Helpline" << endl;

                    cout << "\n\n\t\t\tPress 7 E-Ticket" << endl;

                    cout << "\n\n\t\t\tPress 8 Remove Added Trains" << endl;

                    cout << "\n\n\t\t\tPress 9 Logout" << endl;

                    cout << "\n\n\t\t\tPress 10 Exit" << endl;

                    cout << "Enter your choice: ";

                    cin >> choice;

if (cin.fail()) {

*// If the input is not an integer*

    cin.clear();

cin.sync();

    system("cls");

    cout << "Invalid input. Please enter a valid integer.\n";

*// save all information*

} else{

                    if (choice == 1)

                    {

                        system("cls");

                        manageTrains(names, destinations, departureTimes, numTrains, MAX\_TRAINS);

                        trainSaveFile(names, destinations, departureTimes, numTrains);

                        system("cls");

                    }

                    if (choice == 3)

                    {

                        system("cls");

                        dashboard();

                        system("cls");

                    }

                    if (choice == 4)

                    {

                        system("cls");

                        Sleep(800);

                        header();

                     info();

                        clearScreen();

                    }

                    if (choice == 5)

                    {

                        system("cls");

                        rate();

                        system("cls");

                    }

                    if (choice == 6)

                    {

                        system("cls");

                        helpline();

                        system("cls");

                    }

                    if (choice == 7)

                    {

                        system("cls");

                        eticket();

                        system("cls");

                    }

                    if (choice == 8)

                    {

                        system("cls");

                        removeTrain(names, destinations, departureTimes, numTrains);

                         trainSaveFile(names, destinations, departureTimes, numTrains);

                          removeTrainFromFile( trainName);

                    system("cls");

                    }

                    if (choice == 2)

                    {

                        system("cls");

                        viewTrains(names, destinations, departureTimes, numTrains);

                      trainSaveFile(names, destinations, departureTimes, numTrains);

                        clearScreen();

                    }

                    if (choice == 9)

                    {

                        system("cls");

                        clearScreen();

                        system("cls");

                        Menu("Login");

                        system("cls");

                    }

                    if (choice == 10)

                    {

                        system("cls");

                        Exit();

                        trainSaveFile(names, destinations, departureTimes, numTrains);

                    }

                    else

                    {

                        system("cls");

                        cout << "Invalid choice. Please try again.\n";

                        system("cls");

*//////save all information*

                    }

                }

                }

            }

            else if (role == "User")

            {

                clearScreen();

                User();

            }

            else if (role == "undefined")

            {

                system("cls");

                cout << "UnIdentifed! " << endl;

            }

        }

        else if (option == 2)

        {

            system("cls");

           string name;

    string password;

    string role;

    signupHeader();

    Menu("Signup");

    cin.clear();

    cin.sync();

    cout << "\t\t\t\t\tEnter your Name: ";

    getline(cin, name);

    cout << "\t\t\t\t\tEnter your Password: ";

    getline(cin, password);

    cout << "\t\t\t\t\tEnter your Role (Admin or User): ";

    getline(cin, role);

    while (role != "Admin" && role != "User") {

        cout << "\t\t\t\t\tInvalid role! Please enter 'Admin' or 'User': ";

        getline(cin, role);

    }

            bool isValid = Signup(name, password, role, users, passwords, roles, count, userArrSize);

            if (isValid)

            {

                 saveUserRecord(name, password, role);

                cout << "\t\t\tCongratulation Signup Succeessfully as " << role << endl;

            }

            if (!isValid)

            {

                cout << "\t\t\tInvalid ! Username or Password" << endl;

            }

        }

        clearScreen();

    }

    return 0;

}

*// RMS Header*

void header()

{

    system("Color 0A");

    cout << R"(

$$$$$$\  $$$$$$\ $$$$$$\$$\      $$\      $$\ $$$$$$\$$\     $$\       $$\      $$\ $$$$$$\ $$\   $$\ $$$$$$\  $$$$$$\ $$$$$$$$\$$\      $$\$$$$$$$$\$$\   $$\$$$$$$$$\        $$$$$$\$$\     $$\ $$$$$$\$$$$$$$$\$$$$$$$$\$$\      $$\

$$  \_\_$$\$$  \_\_$$\\\_$$  \_$$ |     $$ | $\  $$ $$  \_\_$$\$$\   $$  |      $$$\    $$$ $$  \_\_$$\$$$\  $$ $$  \_\_$$\$$  \_\_$$\$$  \_\_\_\_\_$$$\    $$$ $$  \_\_\_\_\_$$$\  $$ \\_\_$$  \_\_|      $$  \_\_$$\$$\   $$  $$  \_\_$$\\_\_$$  \_\_$$  \_\_\_\_\_$$$\    $$$ |

$$ |  $$ $$ /  $$ | $$ | $$ |     $$ |$$$\ $$ $$ /  $$ \$$\ $$  /       $$$$\  $$$$ $$ /  $$ $$$$\ $$ $$ /  $$ $$ /  \\_\_$$ |     $$$$\  $$$$ $$ |     $$$$\ $$ |  $$ |         $$ /  \\_\_\$$\ $$  /$$ /  \\_\_| $$ |  $$ |     $$$$\  $$$$ |

$$$$$$$  $$$$$$$$ | $$ | $$ |     $$ $$ $$\$$ $$$$$$$$ |\$$$$  /        $$\$$\$$ $$ $$$$$$$$ $$ $$\$$ $$$$$$$$ $$ |$$$$\$$$$$\   $$\$$\$$ $$ $$$$$\   $$ $$\$$ |  $$ |         \$$$$$$\  \$$$$  / \$$$$$$\   $$ |  $$$$$\   $$\$$\$$ $$ |

$$  \_\_$$<$$  \_\_$$ | $$ | $$ |     $$$$  \_$$$$ $$  \_\_$$ | \$$  /         $$ \$$$  $$ $$  \_\_$$ $$ \$$$$ $$  \_\_$$ $$ |\\_$$ $$  \_\_|  $$ \$$$  $$ $$  \_\_|  $$ \$$$$ |  $$ |          \\_\_\_\_$$\  \$$  /   \\_\_\_\_$$\  $$ |  $$  \_\_|  $$ \$$$  $$ |

$$ |  $$ $$ |  $$ | $$ | $$ |     $$$  / \$$$ $$ |  $$ |  $$ |          $$ |\$  /$$ $$ |  $$ $$ |\$$$ $$ |  $$ $$ |  $$ $$ |     $$ |\$  /$$ $$ |     $$ |\$$$ |  $$ |         $$\   $$ |  $$ |   $$\   $$ | $$ |  $$ |     $$ |\$  /$$ |

$$ |  $$ $$ |  $$ $$$$$$\$$$$$$$$\$$  /   \$$ $$ |  $$ |  $$ |          $$ | \\_/ $$ $$ |  $$ $$ | \$$ $$ |  $$ \$$$$$$  $$$$$$$$\$$ | \\_/ $$ $$$$$$$$\$$ | \$$ |  $$ |         \$$$$$$  |  $$ |   \$$$$$$  | $$ |  $$$$$$$$\$$ | \\_/ $$ |

\\_\_|  \\_\_\\_\_|  \\_\_\\_\_\_\_\_\_\\_\_\_\_\_\_\_\_\\_\_/     \\_\_\\_\_|  \\_\_|  \\_\_|          \\_\_|     \\_\_\\_\_|  \\_\_\\_\_|  \\_\_\\_\_|  \\_\_|\\_\_\_\_\_\_/\\_\_\_\_\_\_\_\_\\_\_|     \\_\_\\_\_\_\_\_\_\_\_\\_\_|  \\_\_|  \\_\_|          \\_\_\_\_\_\_/   \\_\_|    \\_\_\_\_\_\_/  \\_\_|  \\_\_\_\_\_\_\_\_\\_\_|     \\_\_|

)";

}

*/// Train Adding function*

void addTrain(string names[], string destinations[], int departureTimes[], int &numTrains, const int MAX\_TRAINS)

{

    header();

   if (numTrains < MAX\_TRAINS) {

        cout << "\nEnter Train Details:\n";

*// Input validation for Train Name*

        cout << "Train Name: ";

        cin.ignore(); *// Ignore newline character in the input buffer*

        getline(cin, names[numTrains]);

*// Input validation for Destination*

        cout << "Destination: ";

        getline(cin, destinations[numTrains]);

*// Input validation for Departure Time*

        cout << "Departure Time (in 24-hour format, e.g., 1400): ";

        while (!(cin >> departureTimes[numTrains]) || departureTimes[numTrains] < 0 || departureTimes[numTrains] > 2359) {

            cout << "Invalid input. Please enter a valid departure time in 24-hour format (e.g., 1400): ";

            cin.clear(); *// Clear the buffer*

cin.sync();

*// Read and discard characters until a newline is encountered*

            while (cin.get() != '\n')

                continue;

        }

        cout << "Train added successfully!\n";

        numTrains++;

    } else {

        cout << "Maximum number of trains reached. Cannot add more trains.\n";

    }

}

*// VIEW ADDED TRAIN Funcitons*

void viewTrains(const string names[], const string destinations[], const int departureTimes[], int numTrains)

{

    header();

    if (numTrains > 0)

    {

        cout << "\n===== List of Added Trains =====\n";

        cout << setw(15) << "Train Name" << setw(15) << "Destination" << setw(20) << "Departure Time\n";

        for (int i = 0; i < numTrains; ++i)

        {

            cout << setw(15) << names[i] << setw(15) << destinations[i] << setw(20) << departureTimes[i] << "\n";

        }

        getch();

    }

    else

    {

        cout << "No trains added yet.\n";

    }

}

*// INFORAMTION funct*

void info(){

    system("cls");

    header();

  cout << "Pakistan Railways is the national, state-owned railway company of Pakistan in Lahore. Founded in 1861 as the North Western State Railway and headquartered in Lahore, it owns 7,490 kilometres of operational track across Pakistan, stretching from Torkham to Karachi, offering both freight and passenger services." << endl;

}

*// Online Ticket*

void eticket()

{

    system("cls");

    const int MAX\_PASSENGERS = 10;

    header();

    cin.clear();

    cin.sync();

    int numTickets;

    cout << "Enter number of tickets: ";

    while (true)

    {

*// Use getline to for spaces*

        string input;

        getline(cin, input);

*// (also use stoi =string to integer)*

        stringstream ss(input);

        if (ss >> numTickets && ss.eof())

        {

*/////if valid input then loop break*

            break;

        }

        else

        {

            cout << "Invalid input. Please enter a valid number of tickets: ";

        }

    }

    string names[MAX\_PASSENGERS];

    string cnics[MAX\_PASSENGERS];

    string locations[MAX\_PASSENGERS];

    int nums[MAX\_PASSENGERS];

    int totalAmounts[MAX\_PASSENGERS];

    cout << "\t\t\t\t\t\tPASSENGERS DETAILS" << endl;

    for (int i = 0; i < numTickets; ++i)

    {

        cout << "Enter Name for Passenger " << i + 1 << ": ";

*//// Use getline to handle spaces in the name and cnic*

        getline(cin, names[i]);

        cout << "Enter CNIC for Passenger " << i + 1 << ": ";

        getline(cin, cnics[i]);

        bool validLocation = false;

        while (!validLocation)

        {

            cout << "Enter Location for Passenger " << i + 1 << ": ";

            cin >> locations[i];

            if (locations[i] != "Lahore" && locations[i] != "Gujranwala" && locations[i] != "Peshawar" && locations[i] != "Karachi" && locations[i] != "Multan")

            {

                cout << "Invalid Location. Please enter a valid location (Lahore, Gujranwala, Peshawar, Karachi, Multan).\n";

            }

            else

            {

                validLocation = true;

            }

        }

        nums[i] = 1;

        if (locations[i] == "Lahore")

        {

            totalAmounts[i] = nums[i] \* 4500;

        }

        else if (locations[i] == "Karachi")

        {

            totalAmounts[i] = nums[i] \* 4200;

        }

        else if (locations[i] == "Peshawar")

        {

            totalAmounts[i] = nums[i] \* 4500;

        }

        else if (locations[i] == "Multan")

        {

            totalAmounts[i] = nums[i] \* 4000;

        }

        else if (locations[i] == "Gujranwala")

        {

            totalAmounts[i] = nums[i] \* 4500;

        }

    }

    displayInformation(names, cnics, locations, nums, totalAmounts, numTickets);

    clearScreen();

}

*// Display all information*

void displayInformation(string names[], string cnics[], string locations[], int nums[], int totalAmounts[], int numTickets)

{

    clearScreen();

    header();

    cout << "\n\n\n\n\t\t\t\t\t\tTicket Details\n\n\n\n"

         << endl;

    for (int i = 0; i < numTickets; ++i)

    {

        cout << "Passenger " << i + 1 << " Name: " << names[i] << endl;

        cout << "Passenger " << i + 1 << " CNIC: " << cnics[i] << endl;

        cout << "Passenger " << i + 1 << " Location: " << locations[i] << endl;

        cout << "Passenger " << i + 1 << " Number of Tickets: " << nums[i] << endl;

        cout << "Passenger " << i + 1 << " Total Amount: " << totalAmounts[i] << endl;

        cout << "-----------------------------------------" << endl;

    }

}

*// User Menu*

void User()

{

     userHeader();

    int option;

while(true){

    cout << "\n\n\t\t\tPress 1 Home" << endl;

    cout << "\n\n\t\t\tPress 2 Train Timings" << endl;

    cout << "\n\n\t\t\tPress 3 Rates" << endl;

    cout << "\n\n\t\t\tPress 4 Info" << endl;

    cout << "\n\n\t\t\tPress 5 EBooking" << endl;

    cout << "\n\n\t\t\tPress 6 Feedback" << endl;

    cout << "\n\n\t\t\tPress 7 Ratings" << endl;

    cout << "\n\n\t\t\tPress 8 Ticket Information" << endl;

    cout << "\n\n\t\t\tPress 9 Helpline" << endl;

    cout << "\n\n\t\t\tPress 10 Logout" << endl;

      cout << "\n\n\t\t\tInvalid Option Return Back to Login Page\n\n\n";

    cout << "Chose an option..  ";

    cin >> option;

    if (cin.fail())

    {

        Sleep(100);

cout<<"Invalid input."<<endl<<"Try Again";

cin.clear();

cin.sync();

    }

    else{

        break;

    }

    system("cls");

}

    if (option == 4)

    {

        info();

        clearScreen();

        User();

    }

    if (option == 10)

    {

        system("cls");

    }

    if (option == 3)

    {

        system("cls");

        rate();

        User();

    }

    if (option == 1)

    {

        services();

        User();

    }

    if (option == 2)

    {

        dashboard();

        User();

    }

    if (option == 5)

    {

        eticket();

        User();

    }

    if (option == 7)

    {

*//rating header*

        system("cls");

        rating();

        User();

    }

    if (option == 8)

    {

        system("cls");

        addTicketInformation();

        clearScreen();

        User();

    }

    if (option == 6)

    {

*//feedback header*

        system("cls");

        feedback();

        User();

    }

    if (option == 9)

    {

        helpline();

        User();

    }

}

*// Helpline funct*

void helpline()

{

    clearScreen();

    header();

    cout << "Call us at following numbers: " << endl;

    cout << "xxxx-xxx-xxxx" << endl;

    cout << "0000-000-0000\n";

    clearScreen();

}

*// FeedBack Funct*

void feedback()

{

    cin.ignore();

    string res;

    cout << "Enter Your Feedback about our Services: ";

    getline(cin, res);

       saveFeedbackToFile(res);

    cout << "Thanks for your feedback!" << endl;

    Sleep(300);

    clearScreen();

}

*// Rating Funct*

void rating()

{

    header();

    cout << "\t\t\t\tGive Your Response" << endl;

    cout << "1. Good \n";

    cout << "2. Average\n";

    cout << "3. Worst\n";

    int op;

    cout << "Chose any option.. ";

    cin >> op;

    if (op > 3)

    {

        cout << "Invalid choice.Chose other option!";

    }

    else

        cout << "Thanks for your opinion! " << endl;

    Sleep(300);

    clearScreen();

}

*// dashboard funct*

void dashboard()

{

   int op;

    home();

    cout << "\n\n\t\t\tPress 1 Enter Train:" << endl;

    cout << "\n\n\t\t\tPress 2 Location" << endl;

    cout << "\n\n\t\t\tPress 3 Back" << endl;

    cin >> op;

*// Validate the user input*

    if (op < 1 || op > 3)

    {

        cout << "Invalid option. Please enter a valid option." << endl;

        cin.clear(); *// Clear the error flag*

        cin.sync(); *// Discard invalid input*

        return ;*/// as no return in void function*

    }

    if (op == 1)

    {

        home();

        string name;

        cout << "\n\n\t\t\tName: ";

        cin.clear(); *// Clear the error flag*

        cin.sync(); *// Discard invalid input*

        getline(cin, name);

        if (name == "JAFFAR EXPRESS")

        {

            cout << "\n\n\t\t\tTrain Arrive At 10AM";

        }

        else if (name == "YAAD-GAR-EXPRESS")

        {

            cout << "\n\n\t\t\tTrain is on time. ";

        }

        else if (name == "PAK-TRAIN")

        {

            cout << "\n\n\t\t\tTrain is late due to Rain!";

        }

        else if (name == "GREEN TRAIN")

        {

            cout << "\n\n\t\t\tTrain arrives at 2AM";

        }

        else

        {

            cout << "Invalid Name";

        }

    }

    else if (op == 2)

    {

        home();

        string location;

        cout << "\n\n\t\t\tEnter Location: ";

        cin.clear(); *// Clear the error flag*

        cin.sync(); *// Discard invalid input*

        getline(cin, location);

        if (location == "Lahore")

        {

            cout << "\n\n\t\t\tJAFFAR EXPRESS is coming soon! \n";

            Sleep(800);

            system("cls");

            dashboard();

        }

        else if (location == "Karachi")

        {

            cout << "\n\n\t\t\tYAAD-GAR-EXPRESS is on its Way!\n";

            Sleep(800);

            system("cls");

            dashboard();

        }

        else if (location == "Peshawar")

        {

            cout << "\n\n\t\t\tGREEN TRAIN arrives soon! \n";

            Sleep(800);

            system("cls");

            dashboard();

        }

        else if (location == "Multan")

        {

            cout << "\n\n\t\t\tTrain is OUT OF ORDER! \n";

            Sleep(800);

            system("cls");

            dashboard();

        }

        else

        {

            cout << "Invalid Location";

        }

    }

    cout << endl;

    clearScreen();

}

*// Services funct*

void services()

{

*//// our best trains*

    home();

    cout << "\n\n\t\t\tOur Services:";

    cout << "\n\n\t\t\tJAFFAR EXPRESS";

    cout << "\n\n\t\t\tYAAD-GAR-EXPRESS";

    cout << "\n\n\t\t\tGREEN TRAIN";

    cout << "\n\n\t\t\tPAK-TRAIN \n\n\n\n";

    clearScreen();

}

*// Rates funct*

void rate()

{

    system("cls");

    header();

*/////set width for my schedule and looks attractive*

    cout << setw(70) << setfill('-') << "" << setfill(' ') << endl;

    cout << left << setw(30) << "City" << setw(30) << "Train"

         << "Ticket Rate" << endl;

    cout << setw(30) << "Lahore-to-Karachi" << setw(30) << "JAFFAR EXPRESS"

         << "Rs 4500" << endl;

    cout << setw(30) << "Lahore-to-Multan" << setw(30) << "PAK-TRAIN"

         << "Rs 4500" << endl;

    cout << setw(30) << "Karachi-to-Multan" << setw(30) << "YAAD-GAR-EXPRESS"

         << "Rs 4200" << endl;

    cout << setw(30) << "Karachi-to-Lahore" << setw(30) << "JAFFAR EXPRESS"

         << "Rs 4200" << endl;

    cout << setw(30) << "Multan-to-Lahore" << setw(30) << "GREEN TRAIN"

         << "Rs 4000" << endl;

    cout << setw(30) << "Peshawar-to-Karachi" << setw(30) << "JAFFAR EXPRESS"

         << "Rs 4500" << endl;

    cout << setw(30) << "Peshawar-to-Lahore" << setw(30) << "YAAD-GAR-EXPRESS"

         << "Rs 4500" << endl;

    cout << setw(30) << "Gujranwala-to-Lahore" << setw(30) << "PAK-TRAIN"

         << "Rs 4500" << endl;

    clearScreen();

}

*//my login menu*

int loginMenu() {

    int option;

    cout << "\n\n\t\t\t1. Signin" << endl;

    cout << "\n\n\t\t\t2. Signup" << endl;

    cout << "\n\n\t\t\t3. Exit" << endl;

    cout << "\n\n\t\t\tEnter the Option: ";

*///check if option is not b/w 1-3*

    while (!(cin >> option) || option < 1 || option > 3) {

        cout << "\t\t\t\tInvalid Input! "<< endl;

        cout << "\n\n\t\t\tEnter the Option: ";

        cin.clear(); *// Clear input buffer*

        cin.sync();

    }

    return option;*//give option to login functions*

}

*// Sign In Funct*

string Signin(string name, string password, string users[], string passwords[], string roles[], int count)

{

    for (int i = 0; i < count; i++)

    {

        if (users[i] == name && passwords[i] == password)*/////if name and password of sign up is same*

        {

            return roles[i];

        }

    }

    return "undefined";*/////else*

}

*// Sign Up funct*

bool Signup(string name, string password, string role, string users[], string passwords[], string roles[], int &count, int userArrSize)

{

*// Check if the user already exists*

    for (int i = 0; i < count; ++i)

    {

        if (users[i] == name)

        {

            cout << "\t\t\t\t\tUsername already exists. "<<endl<<"Please enter another username." << endl;

            return false; *// User already exists, signup failed*

        }

    }

*// Check if there is space for a new user because maxusers is 100*

    if (count < userArrSize)

    {

        users[count] = name;

        passwords[count] = password;

        roles[count] = role;

        ++count;

        return true; *// Signup successful*

    }

    else

    {

        return false;

    }

}

*// SUB MENUS*

void Menu(string submenu)

{

    string message = submenu + " Menu";

    cout << "\t\t\t\t---------------------" << endl;

    cout << "\t\t\t\t\t" << message << endl;

    cout << "\t\t\t\t---------------------" << endl;

}

void subMenu(string submenu)

{

    string message = "Main Menu > " + submenu;

    cout << message << endl;

    cout << "---------------------" << endl;

}

*// CLEAR SCREEN funct*

void clearScreen()

*///this will clear the screen*

{

    cout << "Press Any Key to Continue.. " << endl;

    getch();

    system("cls");

}

*/// completely exit from applicaton////*

void Exit()

{

    system("cls");

    while (true)

    {

        break;

    }

    exit(0);

}

*// Manage Trains Display*

void displayMenu()

{

    header();

    cout << "1. Add a Train\n";

    cout << "2. View Added Trains\n";

}

*/// Manage Trains Funct*

void manageTrains(string names[], string destinations[], int departureTimes[], int &numTrains, const int MAX\_TRAINS)

{

*/// this will manage trains and give access to add new train*

    addTrain(names, destinations, departureTimes, numTrains, MAX\_TRAINS);

}

*// Remove Added Train funct*

void removeTrain(string names[], string destinations[], int departureTimes[], int &numTrains)

{

*///give acess to remove added trains*

    if (numTrains > 0)

    {

        viewTrains(names, destinations, departureTimes, numTrains); *// Display the list of trains for reference*

        string trainName,destination;

        cout << "\nEnter the name of the train to remove: ";

        cin.ignore(); *// Ignore newline character in the input buffer*

        getline(cin, trainName);

        cout << "\nEnter the destination of the train to remove: ";

        cin.ignore(); *// Ignore newline character in the input buffer*

        getline(cin, destination);

        bool found = false;

        for (int i = 0; i < numTrains; ++i)

        {

            if (names[i] == trainName||destinations[i]==destination)

            {

*// Shift elements to remove the train*

                for (int j = i; j < numTrains - 1; j++)

                {

                    names[j] = names[j + 1];

                    destinations[j] = destinations[j + 1];

                    departureTimes[j] = departureTimes[j + 1];

                }

*// Decrement the number of trains*

                numTrains--;

                cout << "Train '" << trainName << "' removed successfully!\n";

                found = true;

                break;

            }

        }

        if (!found)

        {

            cout << "Train '" << trainName << "' not found. No changes made.\n";

        }

    }

    else

    {

        cout << "No trains to remove.\n";

    }

    clearScreen();

}

*// Ticket Information*

void addTicketInformation()

{

    const int MAX\_PASSENGERS = 10;

    header();

    int numTickets;

    cout << "Enter number of tickets: ";

    cin >> numTickets;

    string names[MAX\_PASSENGERS];

    string cnics[MAX\_PASSENGERS];

    string locations[MAX\_PASSENGERS];

    int nums[MAX\_PASSENGERS];

    int totalAmounts[MAX\_PASSENGERS];

    cout << "\t\t\t\t\t\tPASSENGERS DETAILS" << endl;

    for (int i = 0; i < numTickets; ++i)

    {

        cout << "Enter Name for Passenger " << i + 1 << ": ";

        cin >> names[i];

        cout << "Enter CNIC for Passenger " << i + 1 << ": ";

        cin >> cnics[i];

       bool validLocation = false;

        while (!validLocation)

        {

            cout << "Enter Location for Passenger " << i + 1 << ": ";

            cin >> locations[i];

            if (locations[i] != "Lahore" && locations[i] != "Gujranwala" && locations[i] != "Peshawar" && locations[i] != "Karachi" && locations[i] != "Multan")

            {

                cout << "Invalid Location. Please enter a valid location (Lahore, Gujranwala, Peshawar, Karachi, Multan).\n";

            }

            else

            {

                validLocation = true;

            }

        }

        nums[i] = 1; *// Assuming each passenger gets one ticket by default*

        if (locations[i] == "Lahore")

        {

            totalAmounts[i] = nums[i] \* 4500;

        }

        else if (locations[i] == "Karachi")

        {

            totalAmounts[i] = nums[i] \* 4200;

        }

        else if (locations[i] == "Peshawar")

        {

            totalAmounts[i] = nums[i] \* 4500;

        }

        else if (locations[i] == "Multan")

        {

            totalAmounts[i] = nums[i] \* 4000;

        }

        else if (locations[i] == "Gujranwala")

        {

            totalAmounts[i] = nums[i] \* 4500;

        }

    }

    displayInformation(names, cnics, locations, nums, totalAmounts, numTickets);

}

*// this function return number of records*

string getField(string record, int field)

{

    int comma = 1;

    string result = "";

    for (int x = 0; x < record.length(); x++)*///.lenght() return the length of array or string*

    {

        if (record[x] == ',') *///if comma exist*

        {

            comma = comma + 1;

        }

        else if (comma == field)

        {

            result = result + record[x];

        }

    }

    return result;

}

*// this function store my railway records*

void saveUserRecord(string name, string password, string role)

{

    fstream file;

    file.open("railwayrecord.txt", ios::app);*///open and app use to append or change data*

    file << name << ',';

    file << password << ',';

    file << role << '\n';

    file.close();

}

*//this function load my records and creates file if not exists*

void loadUserRecords(string users[], string passwords[], string roles[], int &numUsers)

{

    string record = "";

    fstream file;

    numUsers = 0;

    file.open("railwayrecord.txt", ios::in);

    if (!file.is\_open()) {

        cout << "User records file not found. Creating a new file." << endl;

        saveUserRecord("defaultUser", "defaultPassword", "defaultRole");

        file.open("userrecords.txt", ios::in);

    }

    cout << "Loading user records..." << endl;

    while (getline(file, record))

    {

        users[numUsers] = getField(record, 1);

        passwords[numUsers] = getField(record, 2);

        roles[numUsers] = getField(record, 3);

        ++numUsers;

    }

    file.close();

}

*// train record*

void trainSaveFile( string names[],  string destinations[],  int departureTimes[], int numTrains)

{

fstream file;

file.open("t.txt",ios::app);*// app*

for (int i = 0; i < numTrains; ++i)

    {

        file << names[i] << ',' << destinations[i] << ',' << departureTimes[i] << endl;

    }

    file.close();

}

*//Load  Data*

void loadtrainFile(string names[], string destinations[], int departureTimes[], int &numTrains)

{

    string record = "";

    fstream file;

    numTrains = 0;

    file.open("t.txt", ios::in);

    if (!file.is\_open())

    {

        cout << "User records file not found. Creating a new file." << endl;

        trainSaveFile(names, destinations, departureTimes, numTrains);

        file.open("t.txt", ios::in);

    }

    cout << "Loading user records..." << endl;

    while (getline(file, record))

    {

        names[numTrains] = getFields(record, 0); *// Assuming the name is at index 0*

        destinations[numTrains] = getFields(record, 1); *// Assuming the destination is at index 1*

        departureTimes[numTrains] = stoi(getFields(record, 2)); *// Assuming the departure time is at index 2*

        ++numTrains;

        cout<<record;

        cout<<endl;

    }

    file.close();

}

*// Check for comma*

string getFields(const string &record, int index)

{

    stringstream ss(record);

    string field;

    for (int i = 0; i < index; ++i)

    {

        getline(ss, field, ',');

    }

    return field;

}

*//REMOVE ADDED TRAIN*

void removeTrainFromFile(const string &trainName) {

    fstream inFile("t.txt");

    fstream outFile("temp.txt");*//TEMPORARY FILE*

  string record;

    bool trainFound = false;

    while (getline(inFile, record)) {

        if (getFields(record, 0) == trainName) {

            cout << "Train '" << trainName << "' removed successfully!\n";

            trainFound = true;

        } else {

            outFile << record << endl;

        }

    }

    inFile.close();

    outFile.close();

    if (!trainFound) {

        cout << "Train '" << trainName << "' not found in the file.\n";

    } else {

*// Remove the original file*

        remove("t.txt");

*// Rename the temporary file to the original file's name*

        rename("temp.txt", "t.txt");

    }

}

void saveFeedbackToFile(const string& feedback) {

    fstream file;

    file.open("feedback.txt", ios::app); *// Open the file in append mode for changes*

    if (file.is\_open()) {

        file << feedback << endl;

        file.close();

    } else {

        cout << "Error opening the file for feedback." << endl;

    }

}

*// login header*

void loginHeader(){

    system("cls");

     system("Color 04");

cout<<R"( \_\_\_\_\_                                                  \_\_\_\_\_

        ( \_\_\_ )------------------------------------------------( \_\_\_ )

         |   |                                                  |   |

         |   |  /$$        /$$$$$$   /$$$$$$  /$$$$$$ /$$   /$$ |   |

         |   | | $$       /$$\_\_  $$ /$$\_\_  $$|\_  $$\_/| $$$ | $$ |   |

         |   | | $$      | $$  \ $$| $$  \\_\_/  | $$  | $$$$| $$ |   |

         |   | | $$      | $$  | $$| $$ /$$$$  | $$  | $$ $$ $$ |   |

         |   | | $$      | $$  | $$| $$|\_  $$  | $$  | $$  $$$$ |   |

         |   | | $$      | $$  | $$| $$  \ $$  | $$  | $$\  $$$ |   |

         |   | | $$$$$$$$|  $$$$$$/|  $$$$$$/ /$$$$$$| $$ \  $$ |   |

         |   | |\_\_\_\_\_\_\_\_/ \\_\_\_\_\_\_/  \\_\_\_\_\_\_/ |\_\_\_\_\_\_/|\_\_/  \\_\_/ |   |

         |\_\_\_|                                                  |\_\_\_|

        (\_\_\_\_\_)------------------------------------------------(\_\_\_\_\_)

)";

}

*// Sign up Header*

void signupHeader() {

    system("cls");

    cout << R"(

 $$$$$$\  $$$$$$\  $$$$$$\  $$\   $$\       $$\   $$\ $$$$$$$\

$$  \_\_$$\ \\_$$  \_|$$  \_\_$$\ $$$\  $$ |      $$ |  $$ |$$  \_\_$$\

$$ /  \\_\_|  $$ |  $$ /  \\_\_|$$$$\ $$ |      $$ |  $$ |$$ |  $$ |

\$$$$$$\    $$ |  $$ |$$$$\ $$ $$\$$ |      $$ |  $$ |$$$$$$$  |

 \\_\_\_\_$$\   $$ |  $$ |\\_$$ |$$ \$$$$ |      $$ |  $$ |$$  \_\_\_\_/

$$\   $$ |  $$ |  $$ |  $$ |$$ |\$$$ |      $$ |  $$ |$$ |

\$$$$$$  |$$$$$$\ \$$$$$$  |$$ | \$$ |      \$$$$$$  |$$ |

 \\_\_\_\_\_\_/ \\_\_\_\_\_\_| \\_\_\_\_\_\_/ \\_\_|  \\_\_|       \\_\_\_\_\_\_/ \\_\_|

)";

}

*//  Sign in Header*

void signinHeader() {

    system("cls");

    cout << R"(

  /$$$$$$  /$$$$$$  /$$$$$$  /$$   /$$       /$$$$$$ /$$   /$$

 /$$\_\_  $$|\_  $$\_/ /$$\_\_  $$| $$$ | $$      |\_  $$\_/| $$$ | $$

| $$  \\_\_/  | $$  | $$  \\_\_/| $$$$| $$        | $$  | $$$$| $$

|  $$$$$$   | $$  | $$ /$$$$| $$ $$ $$        | $$  | $$ $$ $$

 \\_\_\_\_  $$  | $$  | $$|\_  $$| $$  $$$$        | $$  | $$  $$$$

 /$$  \ $$  | $$  | $$  \ $$| $$\  $$$        | $$  | $$\  $$$

|  $$$$$$/ /$$$$$$|  $$$$$$/| $$ \  $$       /$$$$$$| $$ \  $$

 \\_\_\_\_\_\_/ |\_\_\_\_\_\_/ \\_\_\_\_\_\_/ |\_\_/  \\_\_/      |\_\_\_\_\_\_/|\_\_/  \\_\_/

)";

}

*// Admin Menu Header*

void adminHeader(){

    system("cls");

    cout<<R"(# /\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\

             # \/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/

             # /\  /$$$$$$  /$$$$$$$  /$$      /$$ /$$$$$$ /$$   /$$       /$$      /$$ /$$$$$$$$ /$$   /$$ /$$   /$$            /\

             # \/ /$$\_\_  $$| $$\_\_  $$| $$$    /$$$|\_  $$\_/| $$$ | $$      | $$$    /$$$| $$\_\_\_\_\_/| $$$ | $$| $$  | $$            \/

             # /\| $$  \ $$| $$  \ $$| $$$$  /$$$$  | $$  | $$$$| $$      | $$$$  /$$$$| $$      | $$$$| $$| $$  | $$            /\

             # \/| $$$$$$$$| $$  | $$| $$ $$/$$ $$  | $$  | $$ $$ $$      | $$ $$/$$ $$| $$$$$   | $$ $$ $$| $$  | $$            \/

             # /\| $$\_\_  $$| $$  | $$| $$  $$$| $$  | $$  | $$  $$$$      | $$  $$$| $$| $$\_\_/   | $$  $$$$| $$  | $$            /\

             # \/| $$  | $$| $$  | $$| $$\  $ | $$  | $$  | $$\  $$$      | $$\  $ | $$| $$      | $$\  $$$| $$  | $$            \/

             # /\| $$  | $$| $$$$$$$/| $$ \/  | $$ /$$$$$$| $$ \  $$      | $$ \/  | $$| $$$$$$$$| $$ \  $$|  $$$$$$/            /\

             # \/|\_\_/  |\_\_/|\_\_\_\_\_\_\_/ |\_\_/     |\_\_/|\_\_\_\_\_\_/|\_\_/  \\_\_/      |\_\_/     |\_\_/|\_\_\_\_\_\_\_\_/|\_\_/  \\_\_/ \\_\_\_\_\_\_/             \/

             # /\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\

             # \/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/)";

}

*// User Menu Header*

void userHeader(){

    system("cls");

    cout<<R"( .--..--..--..--..--..--..--..--..--..--..--..--..--..--..--..--..--..--..--..--..--..--..--..--..--..--.

            / .. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \

            \ \/\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ \/ /

            \/ /`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'\/ /

            / /\                                                                                                / /\

            / /\ \                                                                                              / /\ \

            \ \/ /                                                                                              \ \/ /

            \/ /                                                                                                \/ /

            / /\                                                                                                / /\

            / /\ \   \_\_    \_\_   \_\_\_\_\_\_   \_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_         \_\_       \_\_  \_\_\_\_\_\_\_\_  \_\_    \_\_  \_\_    \_\_    / /\ \

            \ \/ /  |  \  |  \ /      \ |        \|       \       |  \     /  \|        \|  \  |  \|  \  |  \   \ \/ /

            \/ /   | $$  | $$|  $$$$$$\| $$$$$$$$| $$$$$$$\      | $$\   /  $$| $$$$$$$$| $$\ | $$| $$  | $$    \/ /

            / /\   | $$  | $$| $$\_\_\_\$$| $$\_\_    | $$\_\_| $$      | $$$\ /  $$$| $$\_\_    | $$$\| $$| $$  | $$    / /\

            / /\ \  | $$  | $$ \$$    \ | $$  \   | $$    $$      | $$$$\  $$$$| $$  \   | $$$$\ $$| $$  | $$   / /\ \

            \ \/ /  | $$  | $$ \_\$$$$$$\| $$$$$   | $$$$$$$\      | $$\$$ $$ $$| $$$$$   | $$\$$ $$| $$  | $$   \ \/ /

            \/ /   | $$\_\_/ $$|  \\_\_| $$| $$\_\_\_\_\_ | $$  | $$      | $$ \$$$| $$| $$\_\_\_\_\_ | $$ \$$$$| $$\_\_/ $$    \/ /

            / /\    \$$    $$ \$$    $$| $$     \| $$  | $$      | $$  \$ | $$| $$     \| $$  \$$$ \$$    $$    / /\

            / /\ \    \$$$$$$   \$$$$$$  \$$$$$$$$ \$$   \$$       \$$      \$$ \$$$$$$$$ \$$   \$$  \$$$$$$    / /\ \

            \ \/ /                                                                                              \ \/ /

            \/ /                                                                                                \/ /

            / /\                                                                                                / /\

            / /\ \                                                                                              / /\ \

            \ \/ /                                                                                              \ \/ /

            \/ /                                                                                                \/ /

            / /\.--..--..--..--..--..--..--..--..--..--..--..--..--..--..--..--..--..--..--..--..--..--..--..--./ /\

            / /\ \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \.. \/\ \

            \ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `'\ `' /

            `--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--'`--' )";

}

*//Home Header*

void home(){

    system("cls");

    cout<<R"(~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

            ~                                           ~

            ~                                           ~

            ~                                           ~

            ~                                           ~

            ~ \_\_    \_\_   \_\_\_\_\_\_   \_\_       \_\_  \_\_\_\_\_\_\_\_ ~

            ~|  \  |  \ /      \ |  \     /  \|        \~

            ~| $$  | $$|  $$$$$$\| $$\   /  $$| $$$$$$$$~

            ~| $$\_\_| $$| $$  | $$| $$$\ /  $$$| $$\_\_    ~

            ~| $$    $$| $$  | $$| $$$$\  $$$$| $$  \   ~

            ~| $$$$$$$$| $$  | $$| $$\$$ $$ $$| $$$$$   ~

            ~| $$  | $$| $$\_\_/ $$| $$ \$$$| $$| $$\_\_\_\_\_ ~

            ~| $$  | $$ \$$    $$| $$  \$ | $$| $$     \~

            ~ \$$   \$$  \$$$$$$  \$$      \$$ \$$$$$$$$~

            ~                                           ~

            ~                                           ~

            ~                                           ~

            ~                                           ~

            ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~)";

}

# 2.1 Functions Working Flow

user

admin

main ()

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Addnew** |  | **View add** |  | **Available** |  | **Info()** |  | **Ticket** |  | **Helpline()** | **E-ticket()** |  | **Remove added** |  | **Exit()** |  | **log out()** |
| **trains()** |  | **trains()** |  | **Trains()** |  |  |  | **information()** |  |  |  |  | **trains()** |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Home()** |  | **Train** |  |  |  | **Info()** |  | **e-booking()** |  | **Feedback()** |  | **Ratings()** |
|  |  | **timings()** |  | **rates()** |  |  |  |  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Ticketinformation()** | **Helpline()** | **log out()** |

# 

# 2.2 Data Structure

1. // My User Menu
2. void User();
3. // SERVICES
4. void services();
5. // Rates
6. void rate();
7. // Menu
8. void dashboard();
9. /// Header
10. void header();
11. // E-Ticket
12. void eticket();
13. // Feedback;
14. void feedback();
15. /// User Rating
16. void rating();
17. // Information
18. void info();
19. // Helpline
20. void helpline();
21. // SUB MENU
22. void Menu(string submenu);
23. void subMenu(string submenu);
24. // ADMIN FUNCT
25. /// ADD information
26. void addTicketInformation();
27. /// Displaying Menu
28. void displayMenu();
29. /// View Added Trains By Admin
30. void viewTrains(const string names[], const string destinations[], const int departureTimes[], int numTrains);
31. // Display all user Information
32. void displayInformation(string names[], string cnics[], string locations[], int nums[], int totalAmounts[], int numTickets);
33. // Manage all Trains
34. void manageTrains(string names[], string destinations[], int departureTimes[], int &numTrains, const int MAX\_TRAINS);
35. // Remove Added Trains
36. void removeTrain(string names[], string destinations[], int departureTimes[], int &numTrains);
37. // Train Added by Admin
38. void addTrain(string names[], string destinations[], int departureTimes[], int &numTrains, const int MAX\_TRAINS);
39. /// File Handling
40. // Load Login Information
41. void loadUserRecords(string users[], string passwords[], string roles[], int &numUsers);
42. // Save Login Information
43. void saveUserRecord(string name, string password, string role);
44. // CSV fom Data
45. string getField(string record, int field);
46. string getFields(const string &record, int index);
47. // New File For more Data
48. void loadtrainFile(string names[], string destinations[], int departureTimes[], int &numTrains);
49. void trainSaveFile(string names[], string destinations[], int departureTimes[], int numTrains);
50. void removeTrainFromFile(const string &trainName);
51. // Login
52. int loginMenu();
53. // Sign in Function
54. string Signin(string name, string password, string users[], string passwords[], string roles[], int count);
55. // Sign Up function
56. bool Signup(string name, string password, string role, string users[], string passwords[], string roles[], int &usersCount, int userArrSize);
57. /// HEADERS
58. // LOGIN HEADER
59. void loginHeader();
60. // SIGN UP HEADER
61. void signupHeader();
62. /// SIGN IN HEADER
63. void signinHeader();
64. // ADMIN Header
65. void adminHeader();
66. /// USER Header
67. void userHeader();
68. /// Home HEADER
69. void home();
70. // Other Functions For Clear the Screen And Exit the program
71. void clearScreen();
72. void Exit();
73. // file handling for feedback
74. void saveFeedbackToFile(const string &res);

# 2.3 Weakness in the Application

There exist some major weakness in my application explain below:

## 2.3.1 Account Security

No secure practices for handling sensitive information, such as passwords. Consider encrypting passwords and use secure methods for user authentication.

## 2.3.2 Single Responsibility Function

In my application there exist some functions which are not following the Principle of Single Responsibility Function. It can lead to various issues and make the code harder to understand, maintain and extend.

## 2.3.3 Limited Access

Access is very limited to users. They only have limited access to some functions.

## 2.3.4 Not Access of Berth

Access of Berth is not available, this leads passengers not buy a single seat.

## 2.3.5 Class of Passenger

Class or Type of train is not specified. Elite, A.C, or other classes of passengers were not specified.

# 2.4 Future Directions

Make application using a graphical user interface (GUI) library to create a more visually appealing and user-friendly interface. This can enhance the overall user experience

Provide an offline mode for users to view their booked tickets, train schedules, and other relevant information even when they are not connected to the internet. Class and Berth is added in future.

Security of account is increased and ensured in future. This has a major role in application.

In creating this application my logic building increases and from this Programming Fundamental Course I learn many things. I feel like a beginner programmer.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **A-Extensive Evidence** | | **B-Convincing Evidence** | **C-Limited Evidence** | **D-No Evidence** | |
| Documentation  Formatting **Grade:** | All the documentation meets all the criteria. | | Documentation is well formatted but some of the criteria is not fulfilled. | Documentation is required a lot of improvement. | Documentation is not Available | |
| **Documentation Formatting Criteria:** In **Binder**, **Title** Page, **Header**-Footers, Font **Style**, Font **Size** all are all consistence and according to given **guidelines**. Project **Poster** is professionally design and well presented | | | | | | |
| Documentation Contents **Grade:** | Documentation includes all of the criteria. | | Documentation meet more than 80% of the criteria given. | Documentation meet more than 50% of the criteria. | When the documentation meet less than 50% of the criteria. | |
| **Documentation Contents Criteria:** **Title** Page - **Table** of Contents - Project **Abstract** - **Functional** Requirements - **Wire** Frames –**Data Flow** Diagram-**Data** Structure (Arrays)-**Function** Headers and Description -Project **Code.** - **Weakness** in the Project and **Future** Directions. - **Conclusion** and What your **Learn** from the Project and Course and What is your **Future** Planning. | | | | | | |
| Project  Complexity **Grade:** | Project has at least 2 user’s types and each user  has at least 5 functionalities. | | Project complexity meet 80%  criteria given in extensive evidence | Project complexity meet 50%  criteria given in extensive evidence | Project complexity meet less than 50% criteria given in extensive evidence | |
| Code Style **Grade:** | All Code style criteria is followed | | All code style criteria followed but some  improvements required | lot of improvements required in coding style. | **Did not follow** code style, | |
| **Code Style Criteria:**  Consistent code style. Code is well indented. Variable and Function names are well defined. White Spaces are well used. Comments are added. | | | | | | |
| Code  Documentation Mapping **Grade:** | Code and documentation is synchronized. | | Code and documentation does not synchronized at **some** places | Code and documentation does not synchronized at **many** places | Code and documentation **does not** synchronized. | |
| Data Structure  (Arrays) **Grade:** | Data structure is sufficient for the project requirements | | Data Structure is sufficient but require improvement to meet project requirements. | Data structure is not sufficient and need a lot of improvement | Data Structure is not properly identified and declared. | |
| Modularity **Grade:** | Meet all Modularity criteria | | Meet all Modularity criteria  but at some places it is missing | Do not sufficiently meet the modularity criteria. | No modularity or very minimum modularity. | |
| **Modularity criteria:** Functions are defined for each major feature. No Global array is defined. Functions are independent (identify from parameter list and return types). | | | | | | |
| Validations **Grade:** | Validations on all number type inputs are applied | | Validations are applied but at some places it is missing. | Validations are missing at lot of places | No Validations are used | |
| File Handling **Grade:** | Separate files for separate data. Data in csv format | | File handing require some improvements | File handing require a lot of improvements | Not implemented | |
| Aesthetics of the  User Interface **Grade:** | UI is presentable. Proper coloring, Headers and clear screen is done | | UI require some improvements | UI require a lot of improvements | Not implemented | |
| Presentation and  Demo  **Grade:** | Presentation and Demo was 100% working | | Presentation and Demo require some improvements | Presentation and Demo require a lot of improvements | Presentation was not ok and Demo was not working | |
| Student Understanding with the Code.  **Grade:** | Student has complete understanding how the code is working and knows the concept. | | Student has good understand but some place he does not  know the concepts | Student has a very little understand and lack the major concepts. | Student does not have any level of understanding of the code. | |
| **Checked by:** | |  | | | |
| **Comments:** | |  | | | |

**Student Reg. No:** 2023-CS-28  **Student Name:** Mobeen Butt