Stadium Tickets Management System

Session 2023 - 2027

Submitted by:
Hannan Mushtaq 2023-CS-85

Supervised by:
Dr. Awais Hassan

Course:
CSC-102 Programming Fundamentals

Department of Computer Science
University of Engineering and Technology
Lahore Pakistan

Table of Contents

1. Description

The objective of my project **Stadium Tickets Management System** is to solve the problem of buying stadium tickets online. It will be User-friendly application that will be as simple and effective as can be.

Field of Computer Science

This project will use the applications of computer science. The use of C++ language will be in full effect in the back-end.

Results

The results I expect to deliver at the end of my project is the bill for the customer buying tickets. Also, the admin will be able to check how many tickets have been sold.

2. Users of Application

There will be a total of **two users** in my project:

1. Admin:

The admin will be able to manage the matches data, update ticket prices and total tickets. He will also be able to manage the cafetaria of the stadium and the parking areas.

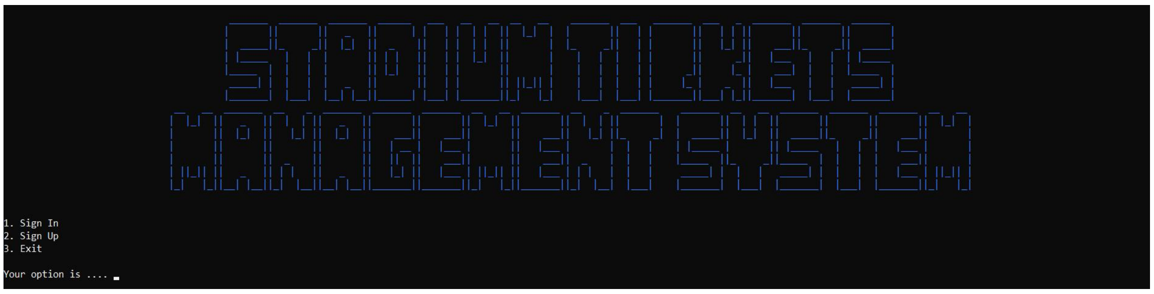
2. Customer:

The customer will be able to buy tickets of his choice and get a receipt. He will also be able to see the cafetaria details and the parking areas around the stadium, give their feedbacks or any complaints regarding our services.

3. Functional Requirements

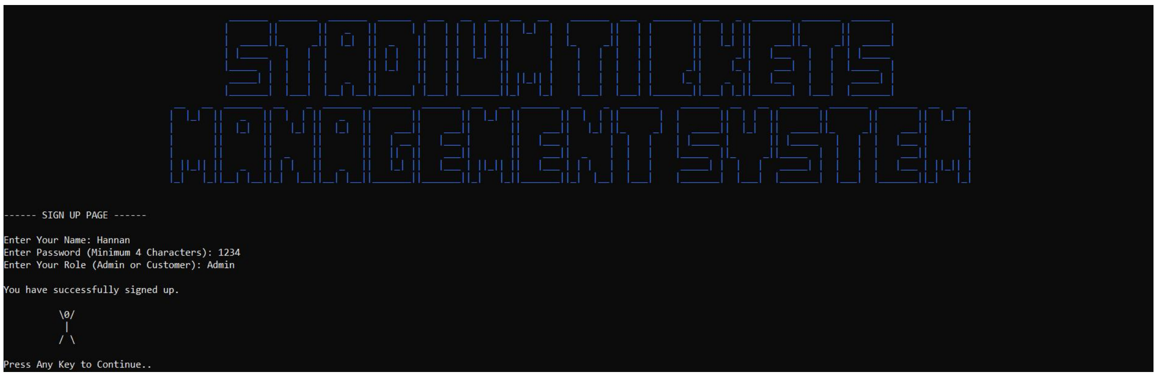
	Functions	So That They Can
Admin	Tickets Data	Update matches and their schedules.
	Sold Tickets Data	See the total Tickets Sold.
	Update Tickets Price	Change the price of the tickets.
	Add Tickets	Add more tickets after construction.
	Manage cafe	Update prices and menu.
	Manage Parking Areas	Update parking areas.
	See Feedbacks	See customer feedbacks.
	See Complaints	See customer Complaints.
	Logout	Logout from the app.
Customer	View Tickets	See ticket prices and types.
	Buy Tickets	Buy tickets of their choice.
	Receipt	Get receipt.
	Checkout	Confirm Purchase.
	Cafe	See menu and prices.
	Parking Areas	See parking areas around the stadium.
	Give Feedback	Give reviews about the services.
	Submit Complaints	Submit any complaints they have.
	Logout	Logout from the app.

4. Wireframes



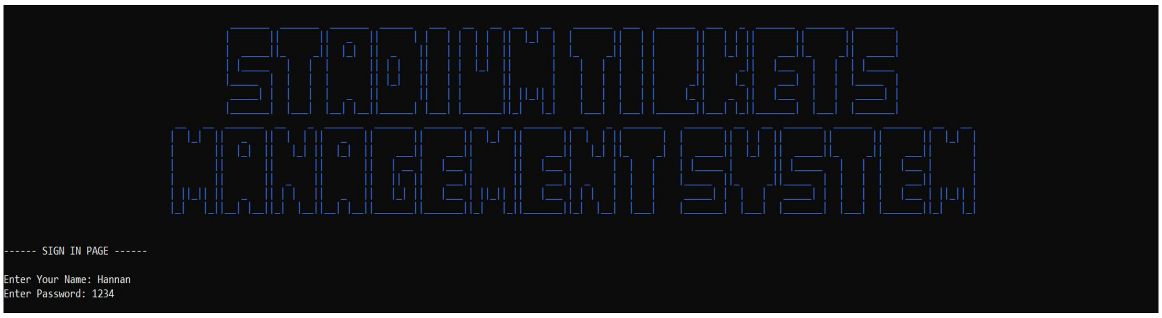
```
1. Sign In
2. Sign Up
3. Exit

Your option is .... 
```

Figure 1: Login Page



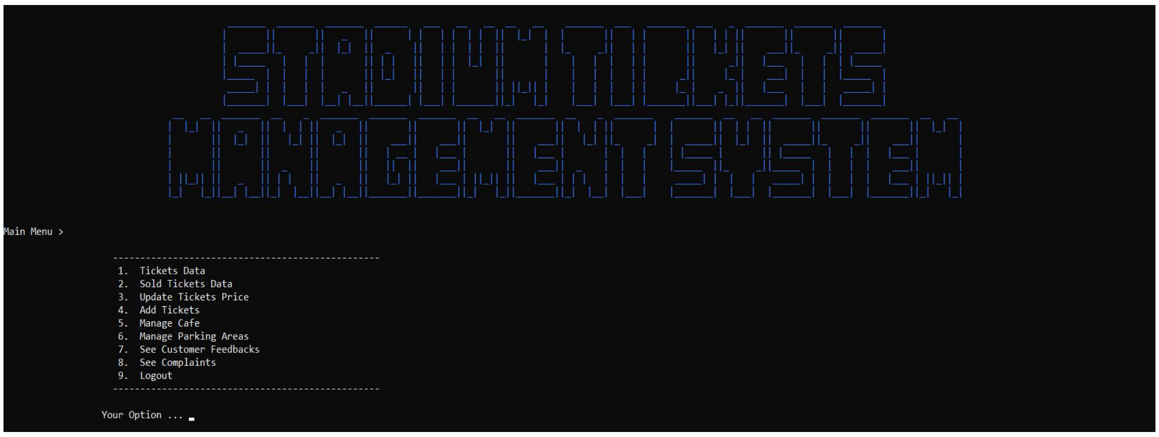
```
------ SIGN UP PAGE ------

Enter Your Name: Hannan
Enter Password (Minimum 4 Characters): 1234
Enter Your Role (Admin or Customer): Admin

You have successfully signed up.

         \0/
          |
         / \

Press Any Key to Continue..
```

Figure 2: Sign Up Page



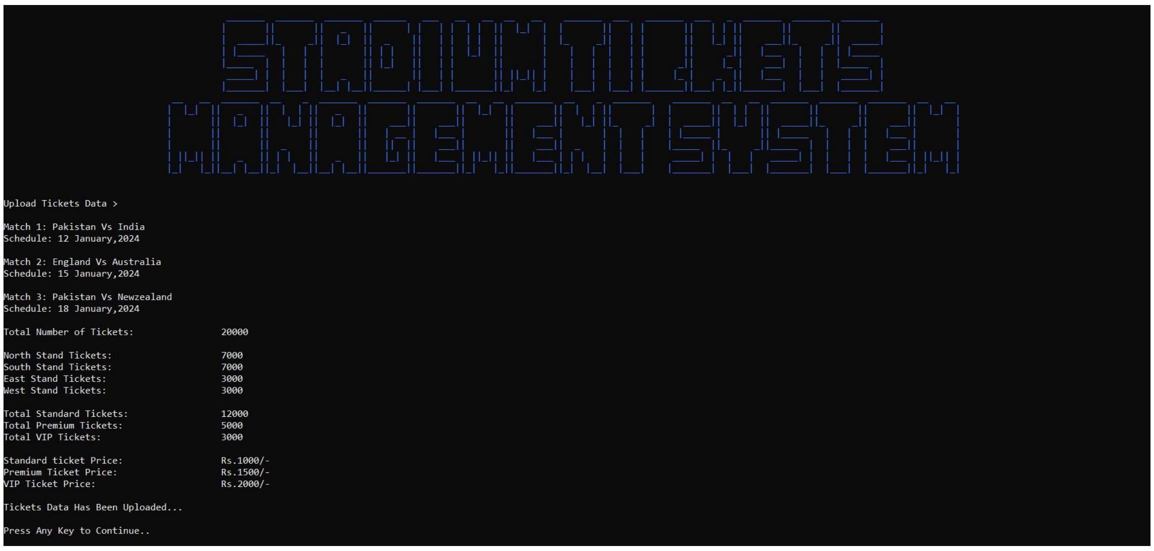
```
------ SIGN IN PAGE ------

Enter Your Name: Hannan
Enter Password: 1234
```

Figure 3: Sign In Page



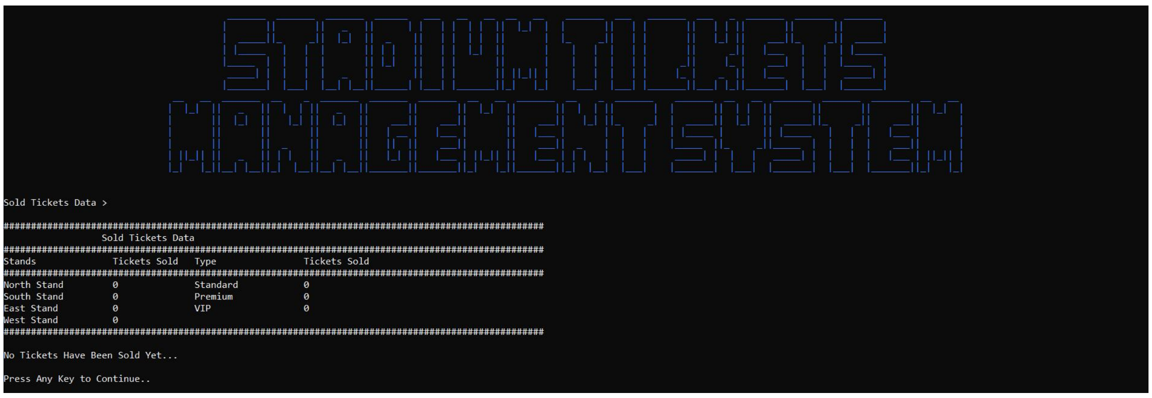
```
Main Menu >

                   -------------------------------------------------
                    1.  Tickets Data
                    2.  Sold Tickets Data
                    3.  Update Tickets Price
                    4.  Add Tickets
                    5.  Manage Cafe
                    6.  Manage Parking Areas
                    7.  See Customer Feedbacks
                    8.  See Complaints
                    9.  Logout
                   -------------------------------------------------

                 Your Option ...
```

Figure 4: Admin Main Menu



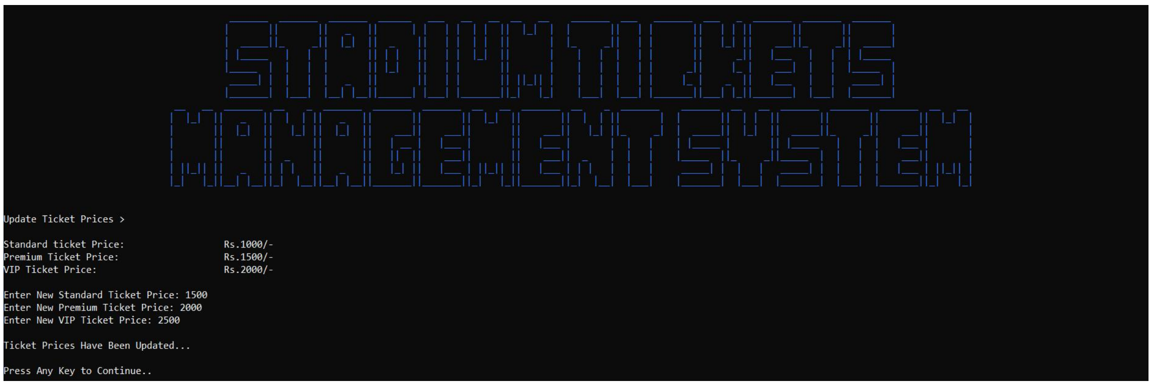
```
Upload Tickets Data >

Match 1: Pakistan Vs India
Schedule: 12 January,2024

Match 2: England Vs Australia
Schedule: 15 January,2024

Match 3: Pakistan Vs Newzealand
Schedule: 18 January,2024

Total Number of Tickets:                20000

North Stand Tickets:                    7000
South Stand Tickets:                    7000
East Stand Tickets:                     3000
West Stand Tickets:                     3000

Total Standard Tickets:                 12000
Total Premium Tickets:                  5000
Total VIP Tickets:                      3000

Standard ticket Price:                  Rs.1000/-
Premium Ticket Price:                   Rs.1500/-
VIP Ticket Price:                       Rs.2000/-

Tickets Data Has Been Uploaded...

Press Any Key to Continue..
```

Figure 5: Upload Tickets Data



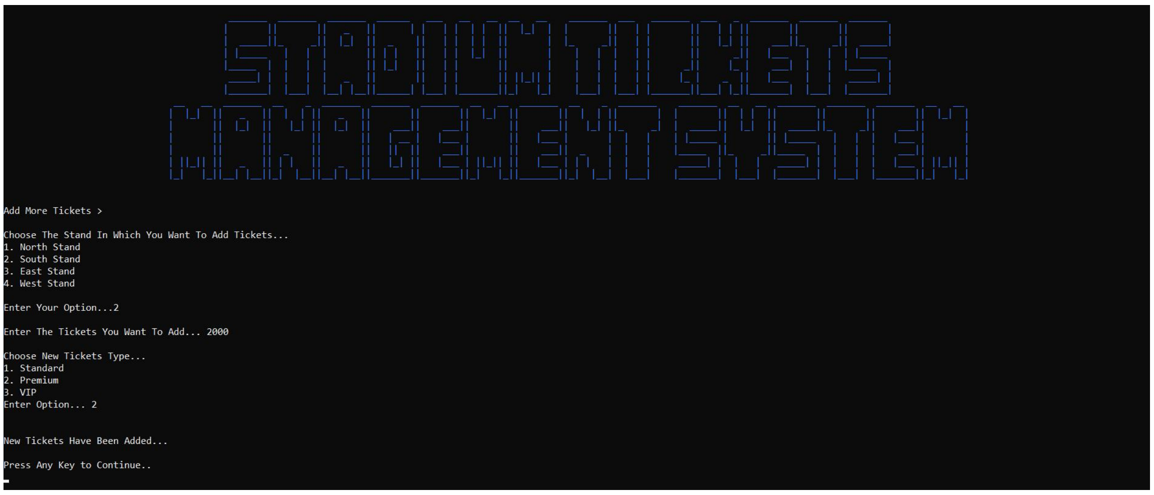
```
Sold Tickets Data >

###################################################################################################
                  Sold Tickets Data
###################################################################################################
Stands              Tickets Sold   Type               Tickets Sold
###################################################################################################
North Stand         0              Standard           0
South Stand         0              Premium            0
East Stand          0              VIP                0
West Stand          0
###################################################################################################

No Tickets Have Been Sold Yet...

Press Any Key to Continue..
```

Figure 6: Sold Tickets Data



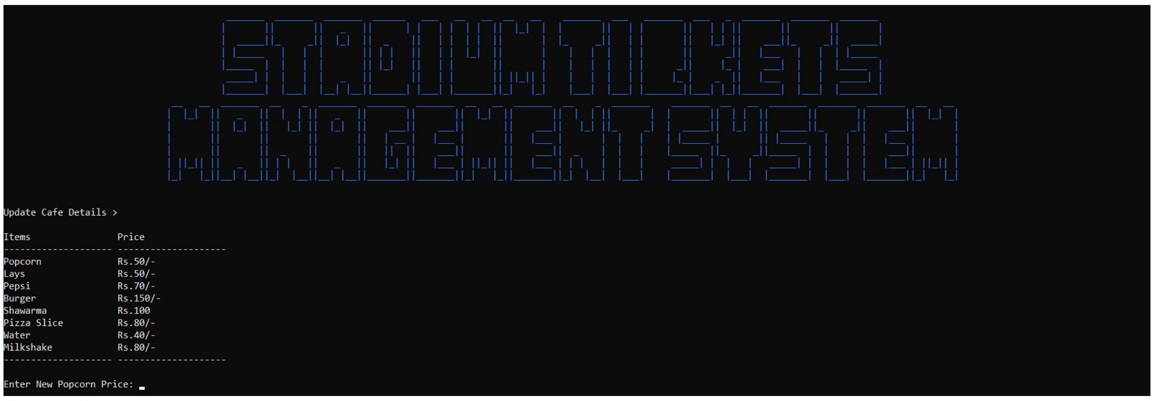
```
Update Ticket Prices >

Standard ticket Price:                  Rs.1000/-
Premium Ticket Price:                   Rs.1500/-
VIP Ticket Price:                       Rs.2000/-

Enter New Standard Ticket Price: 1500
Enter New Premium Ticket Price: 2000
Enter New VIP Ticket Price: 2500

Ticket Prices Have Been Updated...

Press Any Key to Continue..
```

Figure 7: Update Tickets Price



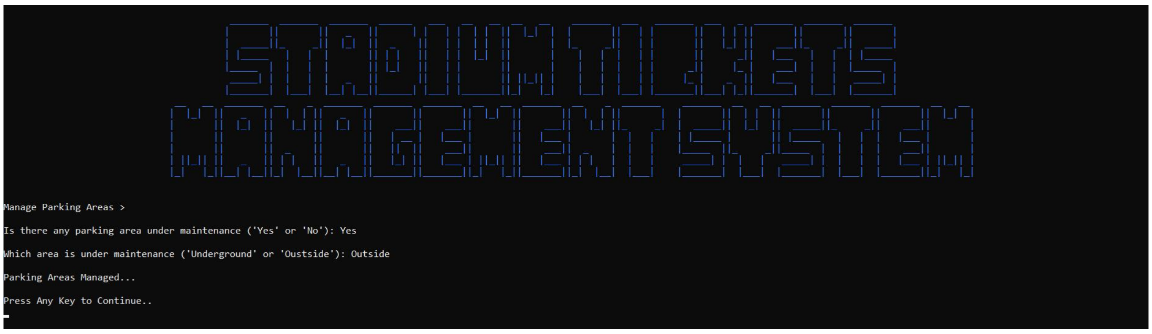
```
Add More Tickets >

Choose The Stand In Which You Want To Add Tickets...
1. North Stand
2. South Stand
3. East Stand
4. West Stand

Enter Your Option...2

Enter The Tickets You Want To Add... 2000

Choose New Tickets Type...
1. Standard
2. Premium
3. VIP
Enter Option... 2


New Tickets Have Been Added...

Press Any Key to Continue..
```

Figure 8: Add More Tickets



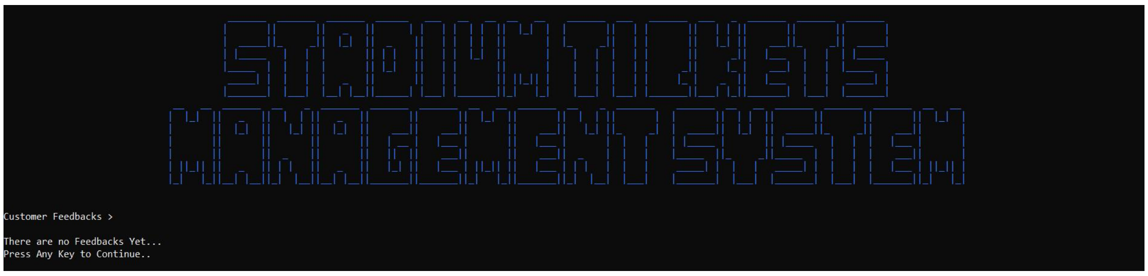
```
Update Cafe Details >

Items                Price
-------------------- --------------------
Popcorn              Rs.50/-
Lays                 Rs.50/-
Pepsi                Rs.70/-
Burger               Rs.150/-
Shawarma             Rs.100
Pizza Slice          Rs.80/-
Water                Rs.40/-
Milkshake            Rs.80/-
-------------------- --------------------

Enter New Popcorn Price:
```

Figure 9: Manage Cafe



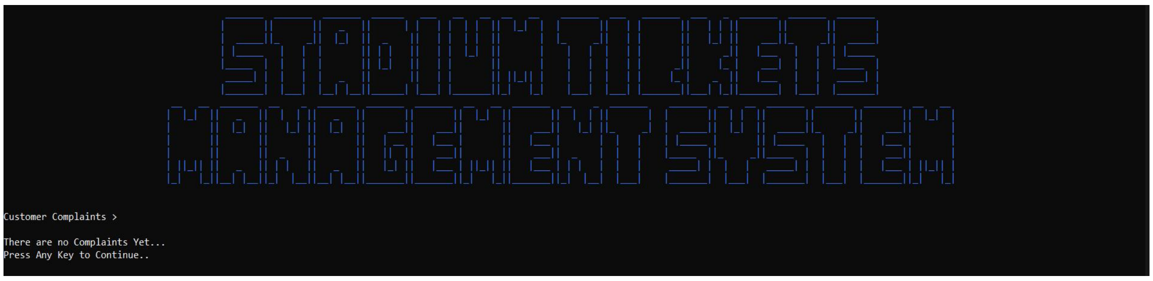
```
Manage Parking Areas >

Is there any parking area under maintenance ('Yes' or 'No'): Yes

Which area is under maintenance ('Underground' or 'Oustside'): Outside

Parking Areas Managed...

Press Any Key to Continue..
```

Figure 10: Manage Parking Areas



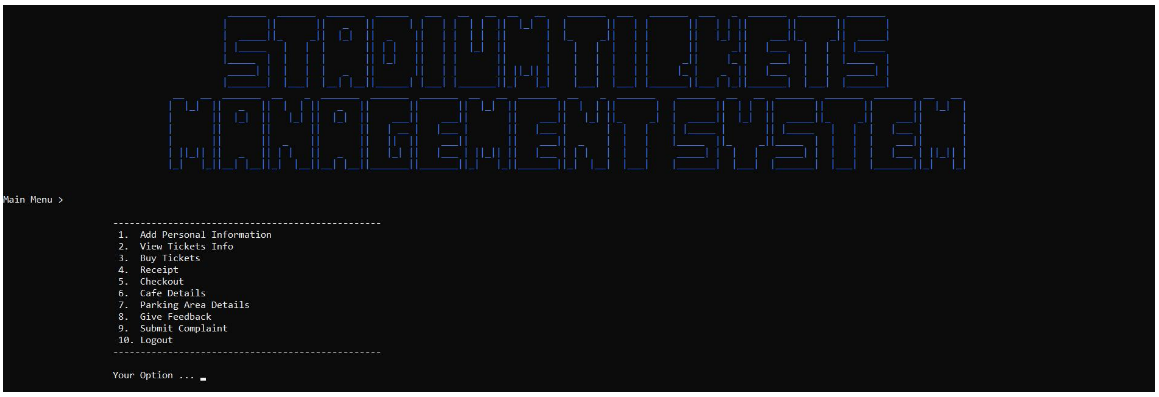
```
Customer Feedbacks >

There are no Feedbacks Yet...
Press Any Key to Continue..
```

Figure 11: View Feedbacks



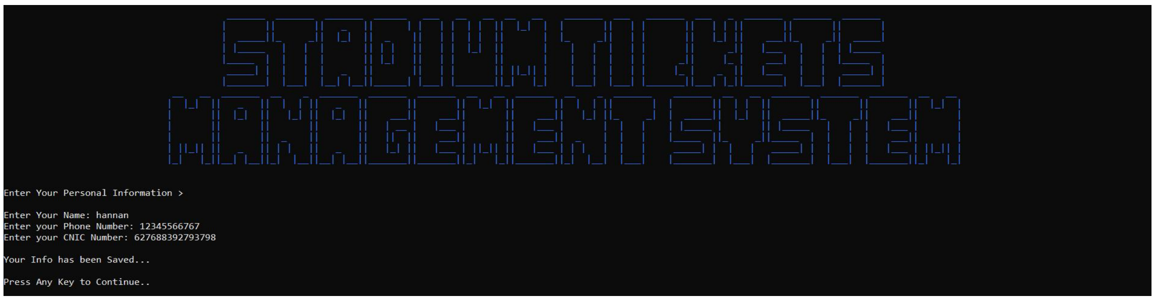
```
Customer Complaints >

There are no Complaints Yet...
Press Any Key to Continue..
```

Figure 12: View Complaints



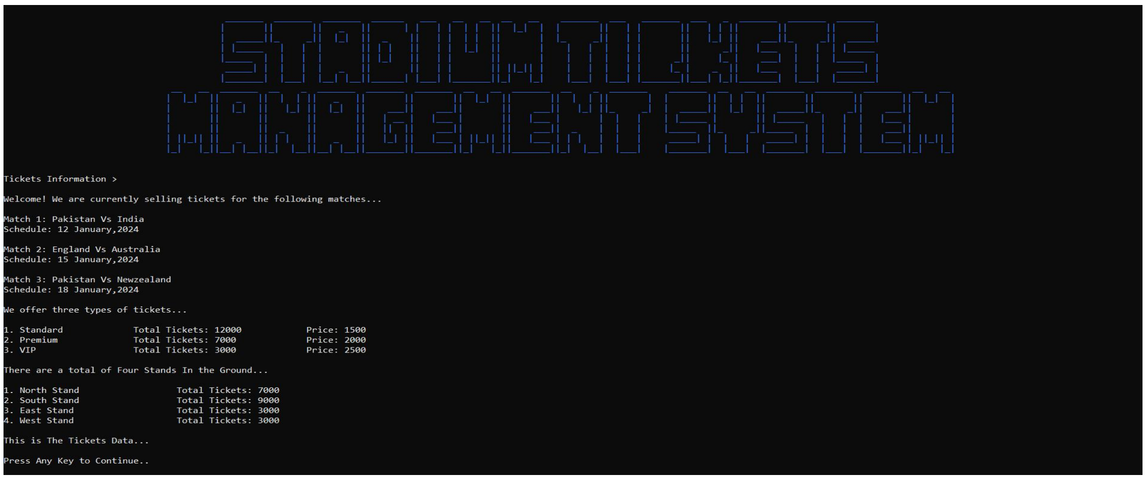
```
Main Menu >

                 ------------------------------------------------
                  1.  Add Personal Information
                  2.  View Tickets Info
                  3.  Buy Tickets
                  4.  Receipt
                  5.  Checkout
                  6.  Cafe Details
                  7.  Parking Area Details
                  8.  Give Feedback
                  9.  Submit Complaint
                  10. Logout
                 ------------------------------------------------

                 Your Option ...
```

Figure 13: Customer Main Menu



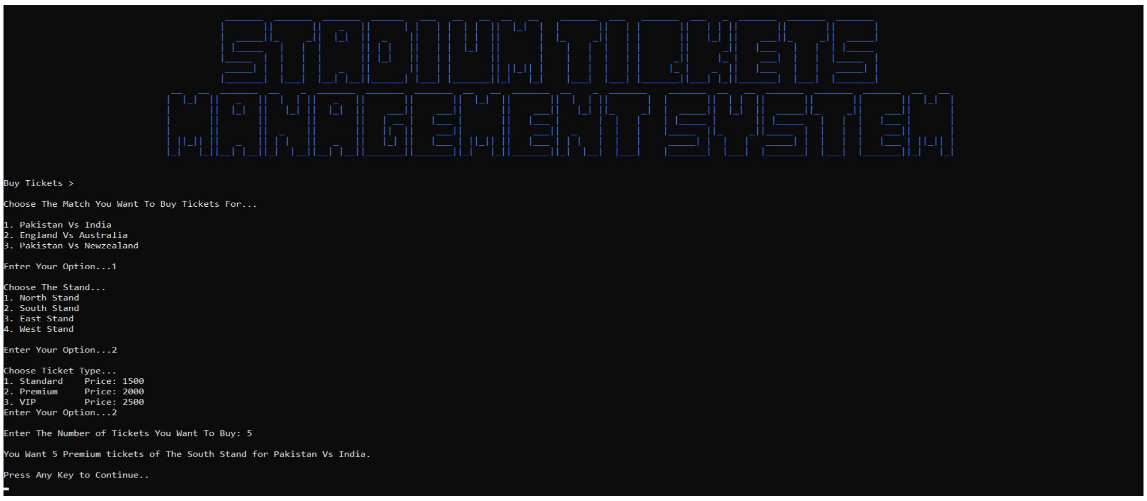
```
Enter Your Personal Information >

Enter Your Name: hannan
Enter your Phone Number: 12345566767
Enter your CNIC Number: 627688392793798

Your Info has been Saved...

Press Any Key to Continue..
```

Figure 14: Add Personal Information



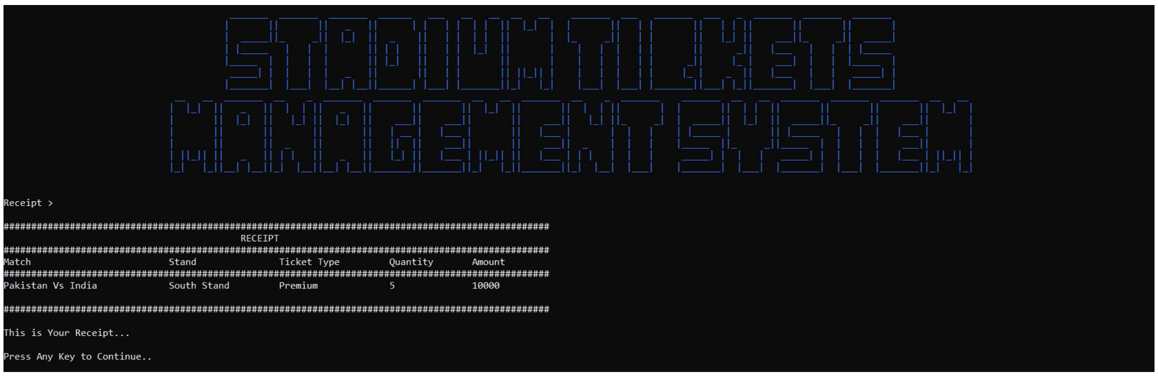
```
Tickets Information >

Welcome! We are currently selling tickets for the following matches...

Match 1: Pakistan Vs India
Schedule: 12 January,2024

Match 2: England Vs Australia
Schedule: 15 January,2024

Match 3: Pakistan Vs Newzealand
Schedule: 18 January,2024

We offer three types of tickets...

1. Standard             Total Tickets: 12000            Price: 1500
2. Premium              Total Tickets: 7000             Price: 2000
3. VIP                  Total Tickets: 3000             Price: 2500

There are a total of Four Stands In the Ground...

1. North Stand                  Total Tickets: 7000
2. South Stand                  Total Tickets: 9000
3. East Stand                   Total Tickets: 3000
4. West Stand                   Total Tickets: 3000

This is The Tickets Data...

Press Any Key to Continue..
```

Figure 15: View Tickets Information



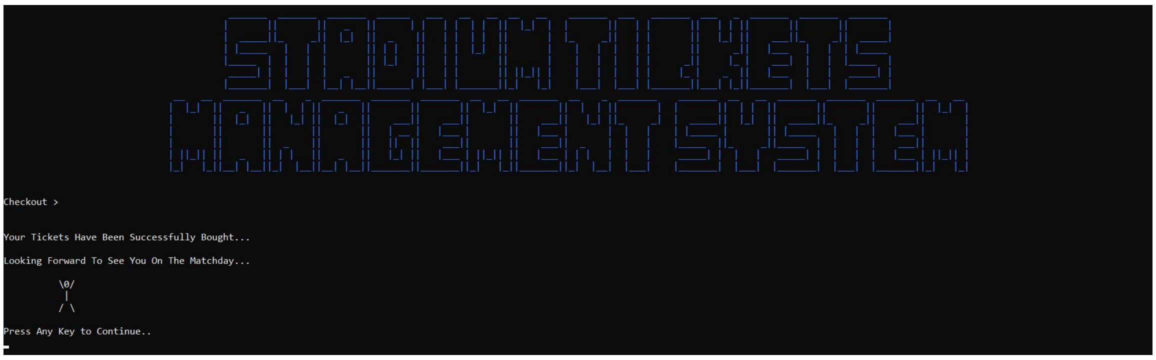
```
Buy Tickets >

Choose The Match You Want To Buy Tickets For...

1. Pakistan Vs India
2. England Vs Australia
3. Pakistan Vs Newzealand

Enter Your Option...1

Choose The Stand...
1. North Stand
2. South Stand
3. East Stand
4. West Stand

Enter Your Option...2

Choose Ticket Type...
1. Standard    Price: 1500
2. Premium     Price: 2000
3. VIP         Price: 2500
Enter Your Option...2

Enter The Number of Tickets You Want To Buy: 5

You Want 5 Premium tickets of The South Stand for Pakistan Vs India.

Press Any Key to Continue..
```

Figure 16: Buy Tickets



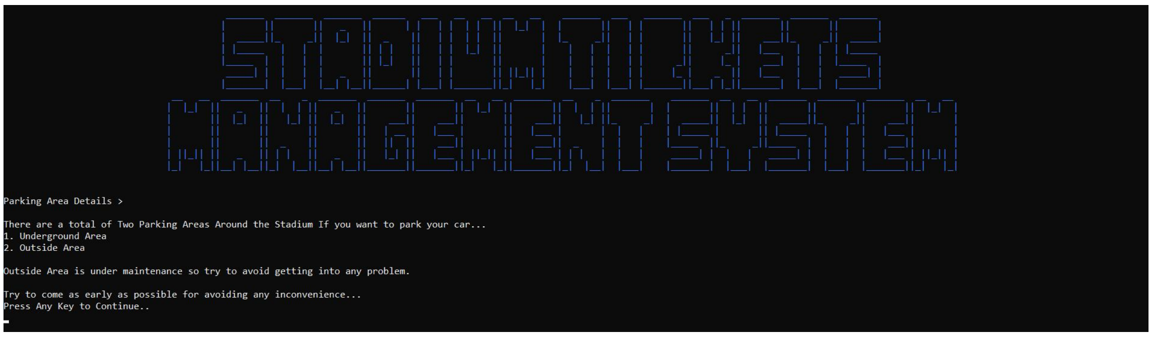
```
Receipt >

####################################################################################################
                                           RECEIPT
####################################################################################################
Match                         Stand               Ticket Type         Quantity       Amount
####################################################################################################
Pakistan Vs India             South Stand         Premium             5              10000

####################################################################################################

This is Your Receipt...

Press Any Key to Continue..
```

Figure 17: Receipt



```
Checkout >


Your Tickets Have Been Successfully Bought...

Looking Forward To See You On The Matchday...

          \0/
           |
          / \

Press Any Key to Continue..
```

Figure 18: Checkout



```
Parking Area Details >

There are a total of Two Parking Areas Around the Stadium If you want to park your car...
1. Underground Area
2. Outside Area

Outside Area is under maintenance so try to avoid getting into any problem.

Try to come as early as possible for avoiding any inconvenience...
Press Any Key to Continue..
```

Figure 19: Parking Area Details

5. Data Structures (Parallel Arrays)

```
const int users = 10000;          // Variable for storing number of users
string name[users];     // Array for storing the usernames of users after signup
string password[users];  // Array for storing the passwords of users after signup
string role[users];       // Array for storing the roles of users after signup
int signupindex = 0;    // Variable for storing index of arrays for signup
string match[3] = {"Pakistan Vs India","England Vs Australia","Pakistan Vs
Newzealand"};           // Array for storing matches data
string schedule[3] = {"12 January","15 January","18 January"};    // Array for storing
schedule data
int tickets = 20000;     // Variable for storing total tickets
int standTicks[4] = {7000,3000,7000,3000};  // Array for storing tickets in each stand
int ticksType[3] = {13000,4000,3000};        // Array for storing tickets of each type
int ticksPrice[3] = {1200,1500,2000}; // Array for storing price of tickets of each type
int ticks = 0;    // Variable for checking whether user has entered ticket details
string feedback[users];          // Array for storing feedbacks given by customers
int feebdbackIndex = 0;          // Variable for storing index of feedback array
string complaint[users];         // Array for storing complaints given by customers
int complaintIndex = 0;          // Variable for storing index of complaints array
int items = 8;                   // Variable for storing total items in cafe
string cafeItems[items] = {"Popcorn","Lays","Pepsi","Burger","Shawarma","Pizza
Slice","Water","Milkshake"};           // Array for storing names of cafe items
int cafeIndex = 0;       // Variable for storing index of arrays in cafe functions
String cafePrice[items] = {"Rs.50/-","Rs.50/-","Rs.70/-","Rs.150/-","Rs.100/-
","Rs.80/-","Rs.40/-","Rs.80/-"};       // Array for storing prices of cafe items
string newCafePrice[items];   // Array for taking input of new prices of cafe items
string parkingAsk;        // Variable for asking admin if there is any parking area under
maintenance
string maintenance;     // Variable for asking admin which area is under maintenance
int park = 0;    // Variable for checking if admin has changed parking area details
string quantity;  // Variable for taking input of quantity of tickets they want to buy
int quantityIndex = 0;           // Variable for storing index of quantity array
int standticksSold[4] = {0,0,0,0};  // Array for storing tickets sold from each stand
string stands[5] = {"","North Stand","South Stand","East Stand","West Stand"};
// Array for storing stand names
string type[4] = {"","Standard","Premium","VIP"};   // Array for storing ticket types
int typeSold[4] = {0,0,0,0};    // Array for storing tickets sold of each type
string askMatchOption;   // Variable for taking input from customer about which
match he wants tickets for
```

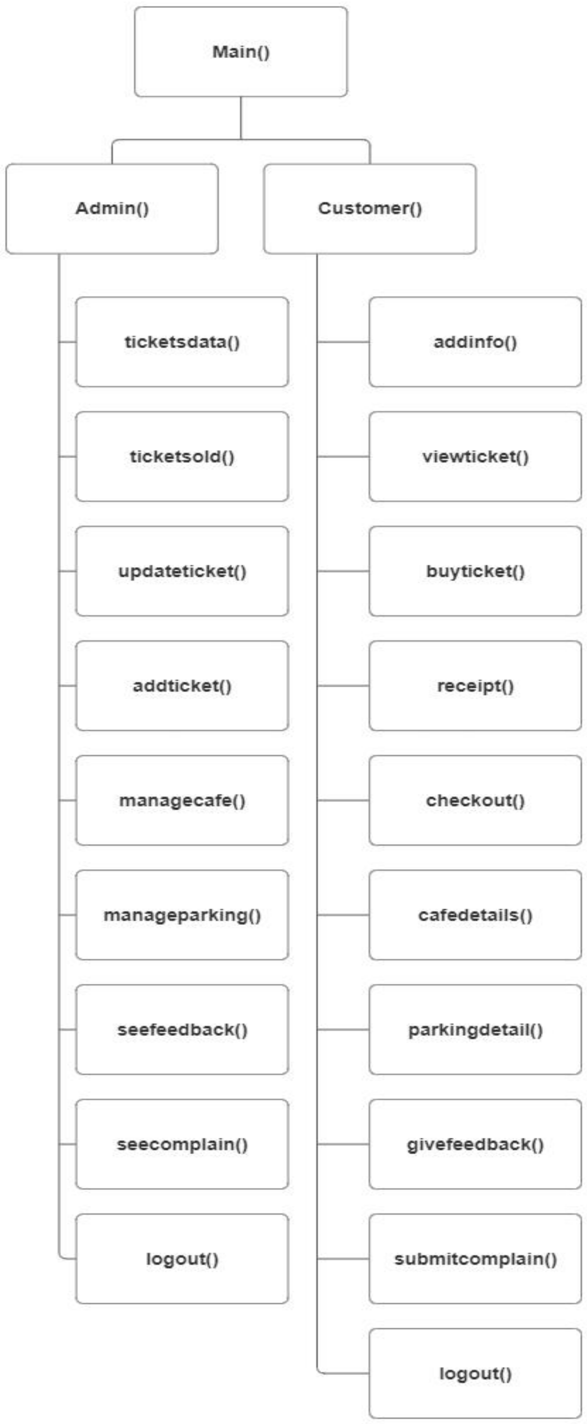
string askStandOption; // Variable for taking input from customer about which stand he wants tickets for
string askTypeOption; // Variable for taking input from customer about which type of tickets he wants
string askMatchIndex; // Variable for storing option returned from askmatch function to use as index for match array
string askStandIndex; // Variable for storing option returned from askstand function to use as index for stand array
string askTypeIndex; // Variable for storing option returned from asktype function to use as index for tickstype array
string addStand; // Variable for asking admin which stand he wants to add tickets to
string addTicks; // Variable for asking admin which stand how many tickets he wants to add
string askAddTypeOption; // Variable for asking admin which type of tickets he wants to add
string newPrice[3]; // Array for taking input of new ticket prices
string filename = "LoginCredentials.txt"; // Variable for storing login credentials data file name
string filename1 = "Matches.txt"; // Variable for storing matches data file name
string filename2 = "Ticket Prices.txt"; // Variable for storing ticket prices data file name
string filename3 = "Total Tickets.txt"; // Variable for storing total tickets data file name
string filename4 = "Feedbacks.txt"; // Variable for storing feedbacks data file name
string filename5 = "Complaints.txt"; // Variable for storing complaints data file name
string filename6 = "Cafe.txt"; // Variable for storing cafe data file name
string option; // Variable for storing option entered by user in login page

6. Function Prototypes

void header();
void clearScreen();
void clearHeader();
void wrongOption();
string getField(string record, int field);
string login();
bool signup(string name[],string password[],string role[],string username,string pass,string roles,int &signupindex,int users,string filename);
void loadLoginCredentialsFile(string filename,string name[],string password[],string role[]);
void readLoginCredentialsFile(string filename,string name[],string password[],string role[],int &signupindex);

```
string signin(string name[],string password[],string role[],string username,string pass,int signupindex);
string adminmenu();
string ticketsData(string match[],string schedule[],int &tickets,int standTicks[],int ticksType[],int ticksPrice[],int &ticks,string filename1);
void updateTicketsData(string match[],string schedule[],int &tickets,int standTicks[],int ticksType[],int ticksPrice[],string filename1);
void ticketsDataOutput(string match[],string schedule[],int &tickets,int standTicks[],int ticksType[],int ticksPrice[]);
void loadMatchesFile(string filename1,string match[],string schedule[]);
void readMatchesFile(string filename1,string match[],string schedule[],int &ticks);
string soldTicketsData(int &ticks,int standticksSold[],string match[],string stands[],string type[],string &askMatchIndex,int typeSold[],int &quantityIndex);
void soldTicketsDataOutput(int standticksSold[],string match[],string stands[],string type[],string &askMatchIndex,int typeSold[]);
void showFeedbacks(string feedback[],int &feedbackIndex);
string addTickets(int &tickets,int standTicks[],int ticksType[],string &addStand,string &addTicks,string &askAddTypeOption,string filename3,int standticksSold[],int typeSold[]);
void addTicketsOutput(int tickets,int standTicks[],int ticksType[]);
void askAddStand(int &tickets,int standTicks[],string &addStand,string &addTicks);
void askAddType(int ticksType[],string &addTicks,string &askAddTypeOption);
void askAddStandOutput(string &addStand);
void askAddTypeOutput(string &askAddTypeOption);
void loadTotalTicketsFile(string filename3,int &tickets,int standTicks[],int ticksType[],int standticksSold[],int typeSold[]);
void readTotalTicketsFile(string filename3,int &tickets,int standTicks[],int ticksType[],int standticksSold[],int typeSold[]);
string updatePrice(string type[],int ticksPrice[],string newPrice[],string filename2);
void updatePriceOutput(string type[],int ticksPrice[],string newPrice[]);
void updatePriceInput(string type[],int ticksPrice[],string newPrice[]);
void loadTicketPricesFile(string filename2,int ticksPrice[]);
void readTicketPricesFile(string filename2,int ticksPrice[]);
void showComplaint(string complaint[],int &complaintIndex);
string cafe(string cafeItems[],string cafePrice[],string newCafePrice[],int &cafeIndex,int &items,string filename6);
void cafeOutput(string cafeItems[],string cafePrice[],string newCafePrice[],int &cafeIndex,int &items);
void loadCafeFile(string filename6,string cafePrice[],int &items);
void readCafeFile(string filename6,string cafePrice[],int &items);
string parkManage(string &parkingAsk,string &maintenance,int &park);
void parkMaintenance(string &parkingAsk,string &maintenance,int &park);
void parkManageArea(string &maintenance,int &park);
string customermenu();
string ticketsInfo(string match[],string schedule[],int &tickets,int standTicks[],int
ticksType[],int ticksPrice[],int &ticks);
void ticketsInfoOutput(string match[],string schedule[],int &tickets,int
standTicks[],int ticksType[],int ticksPrice[],int &ticks);
string giveFeedback(string feedback[],int &feedbackIndex,string filename4);
void giveFeedbackOutput(string feedback[],int &feedbackIndex);
void loadFeedbacksFile(string filename4,string feedback[],int &feedbackIndex);
void readFeedbacksFile(string filename4,string feedback[],int &feedbackIndex);
string giveComplaints(string complaint[],int &complaintIndex,string filename5);
void giveComplaintsOutput(string complaint[],int &complaintIndex);
void loadComplaintsFile(string filename5,string complaint[],int &complaintIndex);
void readComplaintsFile(string filename5,string complaint[],int &complaintIndex);
string showCafe(string cafeItems[],string cafePrice[],int &cafeIndex,int &items);
void showCafeOutput(string cafeItems[],string cafePrice[],int &cafeIndex,int &items);
string parking(string &parkingAsk,string &maintenance,int &park);
void parkingOutput();
void buyTickets(string match[],string schedule[],string &quantity,int ticksPrice[],int
&ticks,string stands[],string type[],int &quantityIndex,int standticksSold[],int
typeSold[],string &askMatchOption,string &askStandOption,string
&askTypeOption,string &askMatchIndex,string &askStandIndex,string
&askTypeIndex,int &tickets,int standTicks[],int ticksType[],string filename3);
string askMatch(string match[],string &askMatchOption);
void askMatchOutput(string match[],string &askMatchOption);
string askStand(string &askStandOption);
void askStandOutput(string &askStandOption);
string askType(int ticksPrice[],string &askTypeOption,string &quantity,int
&quantityIndex,int standticksSold[],string &askStandOption,int typeSold[],int
&ticktets,int standTicks[],int ticksType[]);
void askTypeOutput(int ticksPrice[],string &askTypeOption);
bool quantityCheck(string &quantity);
string receipt(string match[],string stands[],string type[],int ticksPrice[],string
quantity,int &quantityIndex,int &ticks,string &askMatchIndex,string
&askStandIndex,string askTypeIndex);
void receiptOutput(string match[],string stands[],string type[],int ticksPrice[],string
quantity,int &quantityIndex,int &ticks,string &askMatchIndex,string
&askStandIndex,string askTypeIndex);
void checkout(int &quantityIndex);
```

7. Functions Working Flow



8. Complete Code

```
// Libraries

#include<iostream>              // Library for input output functions
#include<conio.h>               // Library for getch() function
#include <iomanip>              // Library for using setw() function
#include <fstream>              // Library for file handling
using namespace std;
```

```
// Function Prototypes
```

```
void header();
void clearScreen();
void clearHeader();
void wrongOption();
string getField(string record, int field);
string login();
bool signup(string name[],string password[],string role[],string username,string pass,string
roles,int &signupindex,int users,string filename);
void loadLoginCredentialsFile(string filename,string name[],string password[],string role[]);
void readLoginCredentialsFile(string filename,string name[],string password[],string
role[],int &signupindex);
string signin(string name[],string password[],string role[],string username,string pass,int
signupindex);
string adminmenu();
string ticketsData(string match[],string schedule[],int &tickets,int standTicks[],int
ticksType[],int ticksPrice[],int &ticks,string filename1);
void updateTicketsData(string match[],string schedule[],int &tickets,int standTicks[],int
ticksType[],int ticksPrice[],string filename1);
void ticketsDataOutput(string match[],string schedule[],int &tickets,int standTicks[],int
ticksType[],int ticksPrice[]);
void loadMatchesFile(string filename1,string match[],string schedule[]);
void readMatchesFile(string filename1,string match[],string schedule[],int &ticks);
string soldTicketsData(int &ticks,int standticksSold[],string match[],string stands[],string
type[],string &askMatchIndex,int typeSold[],int &quantityIndex);
void soldTicketsDataOutput(int standticksSold[],string match[],string stands[],string
type[],string &askMatchIndex,int typeSold[]);
void showFeedbacks(string feedback[],int &feedbackIndex);
string addTickets(int &tickets,int standTicks[],int ticksType[],string &addStand,string
&addTicks,string &askAddTypeOption,string filename3,int standticksSold[],int typeSold[]);
void addTicketsOutput(int tickets,int standTicks[],int ticksType[]);
void askAddStand(int &tickets,int standTicks[],string &addStand,string &addTicks);
void askAddType(int ticksType[],string &addTicks,string &askAddTypeOption);
void askAddStandOutput(string &addStand);
void askAddTypeOutput(string &askAddTypeOption);
void loadTotalTicketsFile(string filename3,int &tickets,int standTicks[],int ticksType[],int
standticksSold[],int typeSold[]);
void readTotalTicketsFile(string filename3,int &tickets,int standTicks[],int ticksType[],int
standticksSold[],int typeSold[]);
string updatePrice(string type[],int ticksPrice[],string newPrice[],string filename2);
void updatePriceOutput(string type[],int ticksPrice[],string newPrice[]);
void updatePriceInput(string type[],int ticksPrice[],string newPrice[]);
void loadTicketPricesFile(string filename2,int ticksPrice[]);
void readTicketPricesFile(string filename2,int ticksPrice[]);
void showComplaint(string complaint[],int &complaintIndex);
string cafe(string cafeItems[],string cafePrice[],string newCafePrice[],int &cafeIndex,int
&items,string filename6);
void cafeOutput(string cafeItems[],string cafePrice[],string newCafePrice[],int &cafeIndex,int
&items);
void loadCafeFile(string filename6,string cafePrice[],int &items);
void readCafeFile(string filename6,string cafePrice[],int &items);
string parkManage(string &parkingAsk,string &maintenance,int &park);
void parkMaintenance(string &parkingAsk,string &maintenance,int &park);
void parkManageArea(string &maintenance,int &park);
string customermenu();
string ticketsInfo(string match[],string schedule[],int &tickets,int standTicks[],int
ticksType[],int ticksPrice[],int &ticks);
void ticketsInfoOutput(string match[],string schedule[],int &tickets,int standTicks[],int
ticksType[],int ticksPrice[],int &ticks);
string giveFeedback(string feedback[],int &feedbackIndex,string filename4);
void giveFeedbackOutput(string feedback[],int &feedbackIndex);
void loadFeedbacksFile(string filename4,string feedback[],int &feedbackIndex);
void readFeedbacksFile(string filename4,string feedback[],int &feedbackIndex);
string giveComplaints(string complaint[],int &complaintIndex,string filename5);
void giveComplaintsOutput(string complaint[],int &complaintIndex);
void loadComplaintsFile(string filename5,string complaint[],int &complaintIndex);
void readComplaintsFile(string filename5,string complaint[],int &complaintIndex);
string showCafe(string cafeItems[],string cafePrice[],int &cafeIndex,int &items);
void showCafeOutput(string cafeItems[],string cafePrice[],int &cafeIndex,int &items);
string parking(string &parkingAsk,string &maintenance,int &park);
void parkingOutput();
void buyTickets(string match[],string schedule[],string &quantity,int ticksPrice[],int
&ticks,string stands[],string type[],int &quantityIndex,int standticksSold[],int
typeSold[],string &askMatchOption,string &askStandOption,string &askTypeOption,string
&askMatchIndex,string &askStandIndex,string &askTypeIndex,int &tickets,int standTicks[],int
ticksType[],string filename3);
string askMatch(string match[],string &askMatchOption);
void askMatchOutput(string match[],string &askMatchOption);
string askStand(string &askStandOption);
void askStandOutput(string &askStandOption);
string askType(int ticksPrice[],string &askTypeOption,string &quantity,int &quantityIndex,int
standticksSold[],string &askStandOption,int typeSold[],int &ticktets,int standTicks[],int
ticksType[]);
void askTypeOutput(int ticksPrice[],string &askTypeOption);
bool quantityCheck(string &quantity);
string receipt(string match[],string stands[],string type[],int ticksPrice[],string
quantity,int &quantityIndex,int &ticks,string &askMatchIndex,string &askStandIndex,string
askTypeIndex);
void receiptOutput(string match[],string stands[],string type[],int ticksPrice[],string
quantity,int &quantityIndex,int &ticks,string &askMatchIndex,string &askStandIndex,string
askTypeIndex);
void checkout(int &quantityIndex);
```

```
// Main Function
```

```
main(){
```

```
    // Declaration of Variables and Arrays
```

```
    const int users = 10000;                        // Variable for storing number of users
    string name[users];                             // Array for storing the usernames of
users after signup
```

```
    string password[users];                         // Array for storing the passwords of
users after signup
    string role[users];                             // Array for storing the roles of users
after signup
    int signupindex = 0;                            // Variable for storing index of arrays
for signup
    string match[3] = {"Pakistan Vs India","England Vs Australia","Pakistan Vs
Newzealand"};         // Array for storing matches data
    string schedule[3] = {"12 January","15 January","18 January"};                      //
Array for storing schedule data
    int tickets = 20000;                            // Variable for storing total tickets
    int standTicks[4] = {7000,3000,7000,3000};      // Array for storing tickets in each stand
    int ticksType[3] = {13000,4000,3000};           // Array for storing tickets of each type
    int ticksPrice[3] = {1200,1500,2000};           // Array for storing price of tickets of
each type
    int ticks = 0;                                  // Variable for checking whether user has
entered ticket details
    string feedback[users];                         // Array for storing feedbacks given by
customers
    int feebdbackIndex = 0;                         // Variable for storing index of feedback
array
    string complaint[users];                        // Array for storing complaints given by
customers
    int complaintIndex = 0;                         // Variable for storing index of
complaints array
    int items = 8;                                  // Variable for storing total items in
cafe
    string cafeItems[items] = {"Popcorn","Lays","Pepsi","Burger","Shawarma","Pizza
Slice","Water","Milkshake"};     // Array for storing names of cafe items
    int cafeIndex = 0;                              // Variable for storing index of arrays in
cafe functions
    string cafePrice[items] = {"Rs.50/-","Rs.50/-","Rs.70/-","Rs.150/-","Rs.100/-","Rs.80/-
","Rs.40/-","Rs.80/-"};      // Array for storing prices of cafe items
    string newCafePrice[items];                     // Array for taking input of new prices of
cafe items
    string parkingAsk;                              // Variable for asking admin if there is
any parking area under maintenance
    string maintenance;                             // Variable for asking admin which area is
under maintenance
    int park = 0;                                   // Variable for checking if admin has
changed parking area details
    string quantity;                                // Variable for taking input of quantity
of tickets they want to buy
    int quantityIndex = 0;                          // Variable for storing index of quantity
array
    int standticksSold[4] = {0,0,0,0};              // Array for storing tickets sold from
each stand
    string stands[5] = {"","North Stand","South Stand","East Stand","West Stand"};          //
Array for storing stand names
    string type[4] = {"","Standard","Premium","VIP"};               // Array for storing
ticket types
    int typeSold[4] = {0,0,0,0};                    // Array for storing tickets sold of each
type
    string askMatchOption;                          // Variable for taking input from customer
about which match he wants tickets for
    string askStandOption;                          // Variable for taking input from customer
about which stand he wants tickets for
    string askTypeOption;                           // Variable for taking input from customer
about which type of tickets he wants
    string askMatchIndex;                           // Variable for storing option returned
from askmatch function to use as index for match array
```

```
    string askStandIndex;                            // Variable for storing option returned
from askstand function to use as index for stand array
    string askTypeIndex;                             // Variable for storing option returned
from asktype function to use as index for tickstype array
    string addStand;                                 // Variable for asking admin which stand
he wants to add tickets to
    string addTicks;                                 // Variable for asking admin which stand
how many tickets he wants to add
    string askAddTypeOption;                         // Variable for asking admin which type of
tickets he wants to add
    string newPrice[3];                              // Array for taking input of new ticket
prices
    string filename = "LoginCredentials.txt";        // Variable for storing login credentials
data file name
    string filename1 = "Matches.txt";                // Variable for storing matches data file
name
    string filename2 = "Ticket Prices.txt";          // Variable for storing ticket prices data
file name
    string filename3 = "Total Tickets.txt";          // Variable for storing total tickets data
file name
    string filename4 = "Feedbacks.txt";              // Variable for storing feedbacks data
file name
    string filename5 = "Complaints.txt";             // Variable for storing complaints data
file name
    string filename6 = "Cafe.txt";                   // Variable for storing cafe data file
name
    string option;                                   // Variable for storing option entered by
user in login page

    // Read Functions calling for storing all data into arrays
```

```
    readLoginCredentialsFile(filename,name,password,role,signupindex);
    readMatchesFile(filename1,match,schedule,ticks);
    readTicketPricesFile(filename2,ticksPrice);
    readTotalTicketsFile(filename3,tickets,standTicks,ticksType,standticksSold,typeSold);
    readFeedbacksFile(filename4,feedback,feebdbackIndex);
    readComplaintsFile(filename5,complaint,complaintIndex);
    readCafeFile(filename6,cafePrice,items);
```

```
    // While Loop for Running the whole system
```

```
    system("cls");
    while (true){

        system("Cls");
        header();
        option = login();
        if (option == "1"){                          // If the user enters option 1 in login
page

            clearHeader();
```

```
            // Taking Input from User During Sign In
```

```
            string username,pass,roles;
            cout <<endl<<"------ SIGN IN PAGE ------"<<endl<<endl;
            cout << "Enter Your Name: ";
            getline(cin>>ws,username);
            cout << "Enter Password: ";
            getline(cin>>ws,pass);
            roles = signin(name,password,role,username,pass,signupindex);

            // If role returned is either 'Admin' or 'admin'
```

```
            if (roles == "Admin" or roles == "admin"){

                // While Loop for running the admin Functionalities
```

```
                while (true){

                    clearHeader();
                    string opt1 = adminmenu();
                    if (opt1 == "1"){                                // If option is '1'
                        clearHeader();
                        string data =
ticketsData(match,schedule,tickets,standTicks,ticksType,ticksPrice,ticks,filename1);
                        cout << data <<endl;
                    }
                    else if (opt1 == "2"){                          // If option is '2'
                        clearHeader();
                        string sold =
soldTicketsData(ticks,standticksSold,match,stands,type,askMatchIndex,typeSold,quantityIndex);
                        cout << sold <<endl;
                    }
                    else if (opt1 == "3"){                          // If option is '3'
                        clearHeader();
                        string update = updatePrice(type,ticksPrice,newPrice,filename2);
                        cout << update <<endl;
                    }
                    else if (opt1 == "4"){                          // If option is '4'
                        clearHeader();
                        string added =
addTickets(tickets,standTicks,ticksType,addStand,addTicks,askAddTypeOption,filename3,standtick
sSold,typeSold);
                        cout << added <<endl;
                    }
                    else if (opt1 == "5"){                          // If option is '5'
                        clearHeader();
                        string cafetaria =
cafe(cafeItems,cafePrice,newCafePrice,cafeIndex,items,filename6);
                        cout << cafetaria <<endl;
                    }
                    else if (opt1 == "6"){                          // If option is '6'
                        clearHeader();
                        string parker = parkManage(parkingAsk,maintenance,park);
                        cout << parker << endl;
                    }
                    else if (opt1 == "7"){                          // If option is '7'
                        clearHeader();
                        showFeedbacks(feedback,feebdbackIndex);
                    }
                    else if (opt1 == "8"){                          // If option is '8'
                        clearHeader();
                        showComplaint(complaint,complaintIndex);
                    }
                    else if (opt1 == "9"){                          // If option is '9'
                        break;
                    }
                    else {                                          // If option is Not Valid
                        cout <<endl<< "Invalid Option Entered! Enter Option Again..." <<endl;
                    }
                    clearScreen();
                }
            }
```

```
            // If role returned is either 'Customer' or 'customer'
```

```
            else if (roles == "Customer" or roles == "customer"){

                // While Loop for running the Customer Functionalities
```

```
                while(true){

                    clearHeader();
                    string opt2 = customermenu();
                    if (opt2 == "1"){                           // If option is '1'
                        clearHeader();
                        string info =
ticketsInfo(match,schedule,tickets,standTicks,ticksType,ticksPrice,ticks);
                        cout << info << endl;
                    }
                    else if (opt2 == "2"){                           // If option is '2'
                        clearHeader();
                        buyTickets(match,schedule,quantity,ticksPrice,ticks,stands,type,quanti
tyIndex,standticksSold,typeSold,askMatchOption,askStandOption,askTypeOption,askMatchIndex,askS
tandIndex,askTypeIndex,tickets,standTicks,ticksType,filename3);
                    }
                    else if (opt2 == "3"){                           // If option is '3'
                        clearHeader();
                        string bill =
receipt(match,stands,type,ticksPrice,quantity,quantityIndex,ticks,askMatchIndex,askStandIndex,
askTypeIndex);
                        cout << bill <<endl;
                    }
                    else if (opt2 == "4"){                           // If option is '4'
                        clearHeader();
                        checkout(quantityIndex);
                    }
                    else if (opt2 == "5"){                           // If option is '5'
                        clearHeader();
                        string item = showCafe(cafeItems,cafePrice,cafeIndex,items);
                        cout << item <<endl;
                    }
                    else if (opt2 == "6"){                           // If option is '6'
                        clearHeader();
                        string par = parking(parkingAsk,maintenance,park);
                        cout << par <<endl;
                    }
                    else if (opt2 == "7"){                           // If option is '7'
                        clearHeader();
                        string feed = giveFeedback(feedback,feebdbackIndex,filename4);
                        cout << feed <<endl;
                    }
                    else if (opt2 == "8"){                           // If option is '8'
                        clearHeader();
                        string comp = giveComplaints(complaint,complaintIndex,filename5);
                        cout << comp <<endl;
                    }
                    else if (opt2 == "9"){                           // If option is '9'
                        break;
                    }
                    else {                                          // If option is Not Valid
                        cout <<endl;
                        cout << "Invalid Option Entered! Enter Option Again..." <<endl;
                    }
                    clearScreen();
                }
            }
```

```
            // If role returned is Invalid

            else if (roles == "invalid"){
                cout <<endl<< "Invalid Credentials! Sign in Again with valid
credentials."<<endl;
            }
        }
        else if (option == "2"){                                    // If the user enters
option '2' in login page

            clearHeader();
            string username,pass,roles;
            cout <<endl<<"------ SIGN UP PAGE ------"<<endl<<endl;
            cout << "Enter Your Name (Without Spaces): ";
            getline(cin>>ws,username);
            cout << "Enter Password (4 Characters): ";
            getline(cin>>ws,pass);
            cout << "Enter Your Role (Admin or Customer): ";
            getline(cin>>ws,roles);
            bool check =
signup(name,password,role,username,pass,roles,signupindex,users,filename);
            if (check == 1){
                cout <<endl<< "You have successfully signed up."<<endl;
                cout << endl;
                cout << "           \\0/"<<endl;
                cout << "            |"<<endl;
                cout << "           / \\"<<endl;
            }
            else if (check == 0){
                cout <<endl<< "Invalid Credentials! Sign Up Again with valid
credentials."<<endl;
            }
        }
        else if (option == "3"){                                    // If the user enters
option '3' in login page
            return 0;
        }
        else {                                                      // If the user enters
invalid option in login page
            cout <<endl<< "Wrong Option Enterd! Enter Option again..."<<endl;

        }
        clearScreen();
    }
}
```

```
void header(){                                  // Function for printing header


    cout <<" \e[1;94m " << R"(
                                         _______  _______  _______  ______   ___   ___   ___  ___
    __    _______  ___   _______  ___   _  _______  _______  _______
                                        |       ||       ||   _   ||      | |   | |   | |
|   ||  |_|  |  |       ||   | |       ||   | |
||       ||       ||       |
                                        |  _____||_     _||  |_|  ||  _    ||   | |   | |
|   ||       |  |_     _||   | |       ||   |_|
||    ___||_     _||  _____|
                                        | |_____   |   |  |       || | |   ||   |
|   |_|  ||       |    |   |  |   | |       ||      _||   |___   |   |  |
|_____
                                        |_____  |  |   |  |       || |_|   ||   |
|       ||       |    |   |  |   | |      _||     |_
|    ___|  |   |  |_____  |
                                         _____| |  |   |  |   _   ||       ||   | |       ||
||_|| |    |   |  |   | |     |_ |    _  ||   |___   |   |   _____|
|
                                        |_______|  |___|  |__| |__||______| |___|
|_______||_|   |_|    |___|  |___| |_______||___|
|_||_______|  |___|  |_______|
                              __   __  _______  __    _  _______  _______  _______  __   __
_______  __    _  _______    _______  __   __  _______  _______  _______  __   __
                              |  |_|  ||   _   ||  |  |
||   _   ||       ||       ||  |_|  ||       ||  |  | ||       |  |       ||  |
|  ||       ||       ||  |_|  |
                              |       ||  |_|  ||   |_|
||  |_|  ||    ___||    ___||       ||    ___||   |_|
||_     _|  |  _____||  |_|  ||  _____||_     _||    ___||        |
                              |       ||       ||       ||       ||   | __ |   |___
|       ||   |___ |       |  |   |    | |_____ |       || |_____   |   |  |   |___ |       |
                              |       ||       ||  _    ||       ||   ||  ||    ___||       ||
    ___||  _    |  |   |    |_____  ||_     _||_____  |  |   |  |    ___||       |
                              | ||_|| ||   _   || | |   ||   _   ||   |_| ||   |___ | ||_||
||   |___ | | |   |  |   |     _____| |  |   |   _____| |  |   |  |   |___ | ||_|| |
                              |_|   |_||__| |__||_|  |__||__|
|__||_______||_______||_|   |_||_______||_|  |__|  |___|    |_______|  |___|  |_______|  |___|
  |_______||_|   |_|
    )"<< "\e[1;37m" << endl;
}
```

```cpp
void clearScreen(){                                  // Function for Clearing screen

    cout <<endl<< "Press Any Key to Continue.." << endl;
    getch();
    system("cls");
}

void clearHeader(){                                  // Function for clearing screen and printing
header

    system("cls");
    header();
}

string login(){                                       // Login Page Function

    string option;
    cout << endl;
    cout << "1. Sign In"<<endl;
    cout << "2. Sign Up"<<endl;
    cout << "3. Exit"<<endl<<endl;
    cout << "Your option is .... ";
    getline(cin>>ws,option);
    return option;
}
```

```cpp
bool signup(string name[],string password[],string role[],string username,string pass,string
roles,int &signupindex,int users,string filename){       // SignUp Function
```

```cpp
    int len = username.length();
    for (int i = 0; i < len; i++)
    {
        if (username[i] == ' ')
        {
            return false;
        }
    }
```

```cpp
    if ((roles != "Admin" && roles != "admin" && roles != "Customer" && roles != "customer")
|| pass.length() != 4){
        return false;
    }
    else if (roles == "Admin" || roles == "admin" || roles == "Customer" || roles ==
"customer"){
```

```cpp
        bool result = false;
        for (int x = 0; x < signupindex; x++){

            if (username == name[x] || pass == password[x]){
                result = true;
                break;
            }
        }
```

```cpp
        if (result == 1){
            return 0;
        }
        else if (signupindex < users){
            name[signupindex] = username;
            password[signupindex] = pass;
            role[signupindex] = roles;
            loadLoginCredentialsFile(filename,name,password,role);
            signupindex++;
            return true;
        }
        else {
            return false;
        }
    }
}
```

```cpp
void loadLoginCredentialsFile(string filename,string name[],string password[],string
role[]){           // Function for storing signup credentials in a file
```

```cpp
    fstream file;
    file.open(filename,ios::out);
    for (int x = 0; x < 90; x++){
        if (name[x] != ""){

            file << name[x] << "," << password[x] << "," << role[x] << endl;
        }
    }
    file.close();
}

// Function for storing signup credentials in their respective arrays from a file

void readLoginCredentialsFile(string filename,string name[],string password[],string
role[],int &signupindex){

    fstream file;
    file.open(filename,ios::in);
    string line;

    while (getline(file, line)){


        if (line != ""){

            name[signupindex] = getField(line, 1);
            password[signupindex] = getField(line, 2);
            role[signupindex] = getField(line, 3);
            signupindex++;
        }
    }
    file.close();
}

string getField(string record,int field){        // Function for returning the required data
from a file depending upon the comma number

    int commaCount = 1;
    string item;
    for (int x = 0; x < record.length(); x++){

        if (record[x] == ','){

            commaCount = commaCount + 1;
        }
        else if (commaCount == field){

            item = item + record[x];
        }
    }
    return item;
}

string signin(string name[],string password[],string role[],string username,string pass,int
signupindex){                // SignIn Function

    for (int x = 0; x < signupindex; x++){

        if (username == name[x] and pass == password[x]){
            return role[x];
        }
    }
    return "invalid";
}
```

```
string customermenu(){                      // Customer Page Function
```

```
    cout << endl;
    cout << "Main Menu >" <<endl<<endl;
    cout << "                        -------------------------------------------------"<<endl;
    cout << "                         1.  View Tickets Info                          "<<endl;
    cout << "                         2.  Buy Tickets                                "<<endl;
    cout << "                         3.  Receipt                                    "<<endl;
    cout << "                         4.  Checkout                                   "<<endl;
    cout << "                         5.  Cafe Details                               "<<endl;
    cout << "                         6.  Parking Area Details                       "<<endl;
    cout << "                         7.  Give Feedback                              "<<endl;
    cout << "                         8.  Submit Complaint                           "<<endl;
    cout << "                         9.  Logout                                     "<<endl;
    cout << "                        -------------------------------------------------"<<endl;
    cout << endl;
    cout << "                         Your Option ... ";
    string option4;
    getline(cin>>ws,option4);
    return option4;
}
```

```
string ticketsInfo(string match[],string schedule[],int &tickets,int standTicks[],int
ticksType[],int ticksPrice[],int &ticks){    // Function for customer to view tickets details

    cout << endl;
    cout << "Tickets Information > " <<endl<<endl;
    if (ticks == 0){
        return "Sorry! The Tickets Data has not been Uploaded By the Admin Yet...";
    }
    else if (ticks > 0){

        ticketsInfoOutput(match,schedule,tickets,standTicks,ticksType,ticksPrice,ticks);
        return "This is The Tickets Data...";
    }
}
```

```
void ticketsInfoOutput(string match[],string schedule[],int &tickets,int standTicks[],int
ticksType[],int ticksPrice[],int &ticks){   // Function for printing output of ticketsInfo
function
```

```
    cout << "Welcome! We are currently selling tickets for the following
matches..."<<endl<<endl;
    for (int x = 0; x < 3; x++){
```

```
        cout << "Match " << x+1 << ": " << match[x] <<endl;
        cout << "Schedule: " << schedule[x] <<endl<<endl;
    }
```

```
    cout << "We offer three types of tickets..."<<endl<<endl;
    cout << "1. Standard   \t\t"  << "Total Tickets: " << ticksType[0] << "\t\tPrice: Rs." <<
ticksPrice[0] << "/-" <<endl;
    cout << "2. Premium    \t\t"  << "Total Tickets: " << ticksType[1] << " \t\tPrice: Rs." <<
ticksPrice[1] << "/-" <<endl;
```

```
    cout << "3. VIP        \t\t"  << "Total Tickets: " << ticksType[2] << " \t\tPrice: Rs." <<
ticksPrice[2] << "/-" <<endl<<endl;
```

```
    cout << "There are a total of Four Stands In the Ground..." <<endl<<endl;
    cout << "1. North Stand  \t\t" << "Total Tickets: " << standTicks[0] <<endl;
    cout << "2. South Stand  \t\t" << "Total Tickets: " << standTicks[1] <<endl;
    cout << "3. East Stand   \t\t" << "Total Tickets: " << standTicks[2] <<endl;
    cout << "4. West Stand   \t\t" << "Total Tickets: " << standTicks[3] <<endl;
    cout <<endl;
}
```

```
string showCafe(string cafeItems[],string cafePrice[],int &cafeIndex,int &items){        //
Function for customer to view the cafe details
```

```
    cout << endl;
    cout << "Cafe Details >" <<endl<<endl;
    showCafeOutput(cafeItems,cafePrice,cafeIndex,items);
    return "These are the Items currently being provided in The Stadium Cafe...";
}
```

```
void showCafeOutput(string cafeItems[],string cafePrice[],int &cafeIndex,int &items){    //
Function for printing cafe details
```

```
    cout << setw(20) << left << "Items" << setw(1) << " " << setw(20) << left << "Price"
<<endl;
        cout << setw(20) << left << "--------------------" << setw(1) << " " << setw(20) <<
left << "--------------------" <<endl;
        for (int x = 0; x < items; x++){
            cout << setw(20) << left << cafeItems[x] << setw(1) << " " << setw(20) << left <<
cafePrice[x] <<endl;
        }
        cout << setw(20) << left << "--------------------" << setw(1) << " " << setw(20) <<
left << "--------------------" <<endl;
        cout <<endl;
}
```

```
string giveFeedback(string feedback[],int &feedbackIndex,string filename4){             //
Function for customer to give his feedback
```

```
    giveFeedbackOutput(feedback,feedbackIndex);
    loadFeedbacksFile(filename4,feedback,feedbackIndex);
    return "Your Feedback has been Submitted...";
}
```

```
void giveFeedbackOutput(string feedback[],int &feedbackIndex){             // Function for
printing output of giveFeedback function

    cout <<endl;
    cout << "Give Feedback >" <<endl<<endl;
    cout << "Give your Feedback About Our Services: ";
    getline(cin>>ws,feedback[feedbackIndex]);
    feedbackIndex++;
    cout <<endl<<endl;
}
```

```
void loadFeedbacksFile(string filename4,string feedback[],int &feedbackIndex){       //
Function for storing feedbacks in a file
```

```
    fstream file;
    file.open(filename4,ios::out);
    for (int x = 0; x < feedbackIndex; x++){

        file << feedback[x] << "," << endl;

    }
    file.close();
}

void readFeedbacksFile(string filename4,string feedback[],int &feedbackIndex){      //
Function for storing feedbacks data into an array from a file

    fstream file;
    file.open(filename4,ios::in);
    string line;

    while (getline(file, line)){


        feedback[feedbackIndex] = getField(line,1);
        feedbackIndex++;
    }
    file.close();



}

string giveComplaints(string complaint[],int &complaintIndex,string filename5){             //
Function for customer to give his complaints

    giveComplaintsOutput(complaint,complaintIndex);
    loadComplaintsFile(filename5,complaint,complaintIndex);
    return "Your Complaint has been Submitted...";
}

void giveComplaintsOutput(string complaint[],int &complaintIndex){          // Function for
printing output of giveComplaints function

    cout <<endl;
    cout << "Submit Complaint >" <<endl<<endl;
    cout << "Give your Complaints About Our Services: ";
    getline(cin>>ws,complaint[complaintIndex]);
    complaintIndex++;
    cout <<endl<<endl;
}

void loadComplaintsFile(string filename5,string complaint[],int &complaintIndex){       //
Function for storing complaints in a file

    fstream file;
    file.open(filename5,ios::out);
    for (int x = 0; x < complaintIndex; x++){

        file << complaint[x] << "," << endl;

    }
    file.close();
}
```

```cpp
void readComplaintsFile(string filename5,string complaint[],int &complaintIndex){       //
Function for storing complaints data into an array from a file

    fstream file;
    file.open(filename5,ios::in);
    string line;

    while (getline(file, line)){

        complaint[complaintIndex] = getField(line,1);
        complaintIndex++;
    }
    file.close();


}

string parking(string &parkingAsk,string &maintenance,int &park){           // Function for
customer to view parking area details

    cout << endl;
    cout << "Parking Area Details > " <<endl<<endl;
    string ans = "Try to come as early as possible for avoiding any inconvenience...";
    if (park == 0){

        parkingOutput();
        return ans;
    }
    else if (park > 0){

        parkingOutput();
        cout << maintenance << " Area is under maintenance so try to avoid getting into any
problem." <<endl<<endl;
        return ans;
    }
}

void parkingOutput(){                                                       // Function for
printing output of parking function

    cout << "There are a total of Two Parking Areas Around the Stadium If you want to park
your car..." <<endl;
    cout << "1. Underground Area" <<endl;
    cout << "2. Outside Area" <<endl<<endl;

}

// Function for customer to buy tickets

void buyTickets(string match[],string schedule[],string &quantity,int ticksPrice[],int
&ticks,string stands[],string type[],int &quantityIndex,int standticksSold[],int
typeSold[],string &askMatchOption,string &askStandOption,string &askTypeOption,string
&askMatchIndex,string &askStandIndex,string &askTypeIndex,int &tickets,int standTicks[],int
ticksType[],string filename3){

    cout << endl;
    cout << "Buy Tickets >" <<endl<<endl;
    if (ticks == 0){
        cout <<  "Sorry! Tickets have not been Uploaded by The Admin Yet...";
    }
    else if (ticks > 0){
```

```
        askMatchIndex = askMatch(match,askMatchOption);
        askStandIndex = askStand(askStandOption);
        askTypeIndex = 
askType(ticksPrice,askTypeOption,quantity,quantityIndex,standticksSold,askStandOption,typeSold,
tickets,standTicks,ticksType);
        cout << endl;
        cout << "You Want " << quantity << " " << type[askTypeIndex[0]-48] << " tickets of The 
" << stands[askStandIndex[0]-48] << " for " << match[askMatchIndex[0]-49] << ".";
        cout << endl;
        loadTotalTicketsFile(filename3,tickets,standTicks,ticksType,standticksSold,typeSold);
    }
}
```

```
string askMatch(string match[],string &askMatchOption){       // Function for asking customer 
which match he wants tickets for
```

```
    while (true){

        askMatchOutput(match,askMatchOption);
        if (askMatchOption == "1" || askMatchOption == "2" || askMatchOption == "3"){
```

```
            return askMatchOption;
            break;
        }
        else {
            wrongOption();
        }
    }
}
```

```
void askMatchOutput(string match[],string &askMatchOption){     // Function for printing 
askMatch output
```

```
    cout << "Choose The Match You Want To Buy Tickets For..." <<endl<<endl;
    cout << "1. " << match[0] <<endl;
    cout << "2. " << match[1] <<endl;
    cout << "3. " << match[2] <<endl<<endl;
    cout << "Enter Your Option...";
    getline(cin>>ws,askMatchOption);
    cout <<endl;
}
```

```
void wrongOption(){                                 // Function for clearing screen and printing 
header if user enters wrong option
```

```
    cout << "Wrong Option Entered..." <<endl;
    clearScreen();
    header();
}
```

```
string askStand(string &askStandOption){            // Function for asking customer which 
stand he wants tickets for
```

```
    while (true){
        askStandOutput(askStandOption);
        if (askStandOption == "1" || askStandOption == "2" || askStandOption == "3" ||
askStandOption == "4"){
            return askStandOption;
            break;
        }
        else {
            wrongOption();
        }
    }
}
```

```
void askStandOutput(string &askStandOption){        // Function for printing askStand function
output
```

```
    cout << "Choose The Stand..." <<endl<<endl;
    cout << "1. North Stand" <<endl;
    cout << "2. South Stand" <<endl;
    cout << "3. East Stand" <<endl;
    cout << "4. West Stand" <<endl<<endl;
    cout << "Enter Your Option...";
    getline(cin>>ws,askStandOption);
    cout << endl;
}
```

```
// Function for asking customer which type of ticket he wants
```

```
string askType(int ticksPrice[],string &askTypeOption,string &quantity,int &quantityIndex,int
standticksSold[],string &askStandOption,int typeSold[],int &tickets,int standTicks[],int
ticksType[]){
```

```
    while(true){
```

```
        askTypeOutput(ticksPrice,askTypeOption);
        if (askTypeOption == "1" || askTypeOption == "2" || askTypeOption == "3"){

            while (true){

                cout << "Enter The Number of Tickets You Want To Buy: ";
                getline(cin>>ws,quantity);
```

```
                if (quantityCheck(quantity)){
```

```
                    standticksSold[askStandOption[0]-49] += stoi(quantity);
                    typeSold[askTypeOption[0]-49] += stoi(quantity);
                    tickets -= stoi(quantity);
                    standTicks[askStandOption[0]-49] -= stoi(quantity);
                    ticksType[askTypeOption[0]-49] -= stoi(quantity);
                    quantityIndex++;
                    break;
                }
                else {
                    wrongOption();
                }
            }
            break;
        }
        else {
            wrongOption();
        }
    }
    return askTypeOption;
}
```

```
bool quantityCheck(string &quantity){           // Function to check whether user has entered 
only numbers during quantity input or not
```

```
    bool result = true;
    int count = 0;
    for (int x = 0; x < quantity.length(); x++){
```

```
        if (quantity[x] == '0' || quantity[x] == '1' || quantity[x] == '2' || quantity[x] == 
'3' || quantity[x] == '5' || quantity[x] == '4' || quantity[x] == '6' || quantity[x] == '7' || 
quantity[x] == '8' || quantity[x] == '9'){

            count++;
        }
    }

    if (count != quantity.length()){
        result = false;
    }
```

```
    return result;
}
```

```
void askTypeOutput(int ticksPrice[],string &askTypeOption){          // Function for printing 
askType function's output
```

```
    cout << "Choose Ticket Type..." <<endl<<endl;
    cout << setw(15) << left << "1. Standard" << "Price: " << ticksPrice[0] <<endl;
    cout << setw(15) << left << "2. Premium" << "Price: " << ticksPrice[1] <<endl;
    cout << setw(15) << left << "3. VIP" << "Price: " << ticksPrice[2] 
<<endl<<endl;
    cout << "Enter Your Option...";
    getline(cin>>ws,askTypeOption);
    cout << endl;
}
```

```
// Function for customer to view a receipt after buying tickets
```

```
string receipt(string match[],string stands[],string type[],int ticksPrice[],string 
quantity,int &quantityIndex,int &ticks,string &askMatchIndex,string &askStandIndex,string 
askTypeIndex){
```

```
    cout << endl;
    cout << "Receipt >" <<endl<<endl;
    if (ticks == 0 || quantityIndex == 0){
        return "Tickets Have not Been Chosen Yet...";
    }
    else if (ticks > 0 && quantityIndex > 0){
```

```
        receiptOutput(match,stands,type,ticksPrice,quantity,quantityIndex,ticks,askMatchIndex,
askStandIndex,askTypeIndex);
        return "This is Your Receipt...";
    }
```

```
}
```

```
// Function for printing output of receipt function
```

```
void receiptOutput(string match[],string stands[],string type[],int ticksPrice[],string
quantity,int &quantityIndex,int &ticks,string &askMatchIndex,string &askStandIndex,string
askTypeIndex){
```

```
    cout << setw(100) << left <<
"#############################################################################################
######" <<endl;
        cout << setw(50) << right << "RECEIPT" << endl;
        cout << setw(100) << left <<
"#############################################################################################
######" <<endl;
        cout << setw(30) << left << "Match" << setw(20) << left << "Stand" << setw(20) << left
<< "Ticket Type" << setw(15) << left << "Quantity" << setw(15) << left << "Amount" <<endl;
        cout << setw(100) << left <<
"#############################################################################################
######" <<endl;
        cout << setw(30) << left << match[askMatchIndex[0]-49] << setw(20) << left <<
stands[askStandIndex[0]-48] << setw(20) << left << type[askTypeIndex[0]-48] << setw(15) <<
left << quantity << setw(20) << left << ticksPrice[askTypeIndex[0]-49]*stoi(quantity) <<endl;
        cout << endl;
        cout << setw(100) << left <<
"#############################################################################################
######" <<endl<<endl;
}
```

```
void checkout(int &quantityIndex){          // Function for customer to view purchase
confirmation
```

```
    cout <<endl<< "Checkout >" <<endl<<endl;
    if (quantityIndex == 0){
        cout << "You Have Not Bought Tickets Yet..." <<endl;
    }
    else if (quantityIndex > 0){

        cout << endl;
        cout << "Your Tickets Have Been Successfully Bought..." <<endl<<endl;
        cout << "Looking Forward To See You On The Matchday..." <<endl<<endl;
        cout << "          \\0/"<<endl;
        cout << "           |"<<endl;
        cout << "          / \\"<<endl;
    }
}
```

```
string adminmenu(){                 // Admin Page Function
```

```
    cout << endl;
    cout << "Main Menu >" <<endl<<endl;
    cout << "                    -------------------------------------------------"<<endl;
    cout << "                     1.  Update Tickets Data                         "<<endl;
    cout << "                     2.  Sold Tickets Data                           "<<endl;
    cout << "                     3.  Update Tickets Price                        "<<endl;
    cout << "                     4.  Add Tickets                                 "<<endl;
    cout << "                     5.  Manage Cafe                                 "<<endl;
    cout << "                     6.  Manage Parking Areas                        "<<endl;
    cout << "                     7.  See Customer Feedbacks                      "<<endl;
    cout << "                      8.  See Complaints                                 "<<endl;
    cout << "                      9.  Logout                                         "<<endl;
    cout << "                     -------------------------------------------------"<<endl;
    cout << endl;
    cout << "                      Your Option ... ";
    string option2;
    getline(cin>>ws,option2);
    return option2;
}
```

```
// Function for Admin to update matches data and view Tickets Information
```

```
string ticketsData(string match[],string schedule[],int &tickets,int standTicks[],int
ticksType[],int ticksPrice[],int &ticks,string filename1){


    while (true){

        cout <<endl;
        cout << "Update Tickets Data >" <<endl<<endl;
        for (int x = 0; x < 3; x++){

            cout << "Match " << x+1 << ": " << match[x] <<endl;
            cout << "Schedule: " << schedule[x] <<endl<<endl;
        }
        string update;
        cout <<endl<< "Do You Want To Update Tickets Data ('Yes' or 'No'): ";
        getline(cin>>ws,update);

        if (update == "Yes" || update == "yes"){

            cout <<endl;
            updateTicketsData(match,schedule,tickets,standTicks,ticksType,ticksPrice,filename1)
;

            loadMatchesFile(filename1,match,schedule);
            return "Tickets Data Has Been Updated... ";
            break;
        }
        else if (update == "No" || update == "no"){

            cout <<endl;
            ticketsDataOutput(match,schedule,tickets,standTicks,ticksType,ticksPrice);
            return "Tickets Data Has Not Been Updated... ";
            break;
        }
        else {
            wrongOption();
        }
    }
}
```

```
// Function for admin to input new matches data
```

```
void updateTicketsData(string match[],string schedule[],int &tickets,int standTicks[],int
ticksType[],int ticksPrice[],string filename1){

    string newMatch[3];
    string newSchedule[3];
    for (int x = 0; x < 3; x++){
```

```cpp
        cout << "Match " << x+1 << ": ";
        getline(cin>>ws,newMatch[x]);
        match[x] = newMatch[x];
        cout << "Schedule: ";
        getline(cin>>ws,newSchedule[x]);
        schedule[x] = newSchedule[x];
        cout <<endl;
    }
    ticketsDataOutput(match,schedule,tickets,standTicks,ticksType,ticksPrice);

}
```

```cpp
void ticketsDataOutput(string match[],string schedule[],int &tickets,int standTicks[],int
ticksType[],int ticksPrice[]){    // Function for printing ticketsData function's output
```

```cpp
    cout << setw(40) << left << "Total Number of Tickets: " << tickets <<endl;
    cout <<endl<< setw(40) << left << "North Stand Tickets: " << standTicks[0] <<endl;
    cout << setw(40) << left << "South Stand Tickets: " << standTicks[1] <<endl;
    cout << setw(40) << left << "East Stand Tickets: " << standTicks[2] <<endl;
    cout << setw(40) << left << "West Stand Tickets: " << standTicks[3] <<endl;
    cout <<endl<< setw(40) << left << "Total Standard Tickets: " << ticksType[0] <<endl;
    cout << setw(40) << left << "Total Premium Tickets: " << ticksType[1] <<endl;
    cout << setw(40) << left << "Total VIP Tickets: " << ticksType[2] <<endl;
    cout <<endl<< setw(40) << left << "Standard ticket Price: " << "Rs." << ticksPrice[0] <<
"/-" <<endl;
    cout << setw(40) << left << "Premium Ticket Price: " << "Rs." << ticksPrice[1] << "/-"
<<endl;
    cout << setw(40) << left << "VIP Ticket Price: " << "Rs." << ticksPrice[2] << "/-" <<endl;
    cout <<endl;
}
```

```cpp
void loadMatchesFile(string filename1,string match[],string schedule[]){        // Function
for storing matches data in a file
```

```cpp
    fstream file;
    file.open(filename1,ios::out);
    for (int x = 0; x < 3; x++){
```

```cpp
        if (match[x] != ""){
```

```cpp
            file << match[x] << "," << schedule[x] << endl;
        }
    }
    file.close();
}
```

```cpp
void readMatchesFile(string filename1,string match[],string schedule[],int &ticks){    //
Function for storing matches data stored in a file into their respective arrays
```

```cpp
    fstream file;
    file.open(filename1,ios::in);
    string line;
    int x = 0;
```

```cpp
    while (getline(file, line)){
```

```cpp


        if (line != ""){
            match[x] = getField(line, 1);
            schedule[x] = getField(line, 2);
            x++;
        }
    }
    file.close();
    ticks++;
}
```

```
// Function for admin to add more tickets
```

```
string addTickets(int &tickets,int standTicks[],int ticksType[],string &addStand,string
&addTicks,string &askAddTypeOption,string filename3,int standticksSold[],int typeSold[]){
```

```
    while (true){

        cout <<endl;
        cout << "Add More Tickets >" <<endl<<endl;
        addTicketsOutput(tickets,standTicks,ticksType);
        cout << "Do You Want To Add More Tickets ('Yes' or 'No'): ";
        string update;
        getline(cin>>ws,update);
        if (update == "Yes" || update == "yes"){
```

```
            cout <<endl;
            askAddStand(tickets,standTicks,addStand,addTicks);
            askAddType(ticksType,addTicks,askAddTypeOption);
            loadTotalTicketsFile(filename3,tickets,standTicks,ticksType,standticksSold,typeSol
d);
            cout <<endl;
            return "New Tickets Have Been Added...";
            break;
        }
        else if (update == "No" || update == "no"){
```

```
            cout <<endl;
            return "New Tickets Have Not Been Added... ";
            break;
        }
        else {
            wrongOption();
        }
    }
}
```

```
void addTicketsOutput(int tickets,int standTicks[],int ticksType[]){            // Function
for viewing the number of tickets of each type
```

```
    cout << setw(40) << left << "Total Number of Tickets: " << tickets <<endl;
    cout <<endl<< setw(40) << left << "North Stand Tickets: " << standTicks[0] <<endl;
    cout << setw(40) << left << "South Stand Tickets: " << standTicks[1] <<endl;
    cout << setw(40) << left << "East Stand Tickets: " << standTicks[2] <<endl;
    cout << setw(40) << left << "West Stand Tickets: " << standTicks[3] <<endl;
    cout <<endl<< setw(40) << left << "Total Standard Tickets: " << ticksType[0] <<endl;
    cout << setw(40) << left << "Total Premium Tickets: " << ticksType[1] <<endl;
    cout << setw(40) << left << "Total VIP Tickets: " << ticksType[2] <<endl<<endl;
}
```

```cpp
void askAddStand(int &tickets,int standTicks[],string &addStand,string &addTicks){          // Function for asking admin which stand he wants to add tickets

    while (true){

        askAddStandOutput(addStand);
        if (addStand == "1" || addStand == "2" || addStand == "3" || addStand == "4"){

            cout << "Enter The Amount of Tickets You Want To Add... ";
            getline(cin>>ws,addTicks);
            tickets += stoi(addTicks);
            standTicks[addStand[0]-49] += stoi(addTicks);
            break;
        }
        else {
            wrongOption();
        }
    }
}

void askAddStandOutput(string &addStand){                                  // Function for printing askAddStand function's output

    cout << "Choose The Stand In Which You Want To Add Tickets..." <<endl;
    cout << "1. North Stand" <<endl;
    cout << "2. South Stand" <<endl;
    cout << "3. East Stand" <<endl;
    cout << "4. West Stand" <<endl<<endl;
    cout << "Enter Your Option...";
    getline(cin>>ws,addStand);
    cout << endl;
}

void askAddType(int ticksType[],string &addTicks,string &askAddTypeOption){          // Function for asking admin which type of tickets he wants to add

    while (true){

        askAddTypeOutput(askAddTypeOption);
        if (askAddTypeOption == "1" || askAddTypeOption == "2" || askAddTypeOption == "3"){

            ticksType[askAddTypeOption[0]-49] += stoi(addTicks);
            break;
        }
        else {
            wrongOption();
        }
    }
}

void askAddTypeOutput(string &askAddTypeOption){                          // Function for printing askAddType function's output

    cout << endl;
    cout << "Choose New Tickets Type... " <<endl;
    cout << "1. Standard" <<endl;
    cout << "2. Premium" <<endl;
    cout << "3. VIP" <<endl;
    cout << "Enter Option... ";
    getline(cin>>ws,askAddTypeOption);
    cout << endl;
}
```

```
// Function for storing tickets data into a file
```

```
void loadTotalTicketsFile(string filename3,int &tickets,int standTicks[],int ticksType[],int
standticksSold[],int typeSold[]){
```

```
    fstream file;
    file.open(filename3,ios::out);
    for (int x = 0; x < 4; x++){
```

```
        file << standTicks[x] << ",";
    }
    for (int x = 0; x < 3; x++){
```

```
        file << ticksType[x] << ",";
    }
    for (int x = 0; x < 4; x++){
```

```
        file << standticksSold[x] << ",";
    }
    for (int x = 0; x < 3; x++){
```

```
        file << typeSold[x] << ",";
    }
    file << tickets;
    file.close();
}
```

```
// Function for storing tickets data stored in a file into their respective arrays
```

```
void readTotalTicketsFile(string filename3,int &tickets,int standTicks[],int ticksType[],int
standticksSold[],int typeSold[]){
```

```
    fstream file;
    file.open(filename3,ios::in);
    string line;
```

```
    while (getline(file, line)){
```

```
        standTicks[0] = stoi(getField(line,1));
        standTicks[1] = stoi(getField(line,2));
        standTicks[2] = stoi(getField(line,3));
        standTicks[3] = stoi(getField(line,4));
        ticksType[0] = stoi(getField(line,5));
        ticksType[1] = stoi(getField(line,6));
        ticksType[2] = stoi(getField(line,7));
        standticksSold[0] = stoi(getField(line,8));
        standticksSold[1] = stoi(getField(line,9));
        standticksSold[2] = stoi(getField(line,10));
        standticksSold[3] = stoi(getField(line,11));
        typeSold[0] = stoi(getField(line,12));
        typeSold[1] = stoi(getField(line,13));
        typeSold[2] = stoi(getField(line,14));
        tickets = stoi(getField(line,15));
```

```
    }
    file.close();

}
```

```
string updatePrice(string type[],int ticksPrice[],string newPrice[],string
filename2){              // Function for admin to update ticket prices
```

```
    while (true){
```

```
        updatePriceOutput(type,ticksPrice,newPrice);
        string update;
        cout << "Do You Want To Update Ticket Prices ('Yes' or 'No'): ";
        getline(cin>>ws,update);
        if (update == "Yes" || update == "yes"){
```

```
            cout <<endl;
            updatePriceInput(type,ticksPrice,newPrice);
            loadTicketPricesFile(filename2,ticksPrice);
            return "Ticket Prices Have Been Updated... ";
            break;
        }
        else if (update == "No" || update == "no"){
```

```
            cout <<endl;
            return "Ticket Prices Have Not Been Updated... ";
            break;
        }
        else {
            wrongOption();
        }
    }
}
```

```
void updatePriceOutput(string type[],int ticksPrice[],string newPrice[]){           //
Function for printing updatePrice function's output
```

```
    cout << endl;
    cout << "Update Ticket Prices > " <<endl<<endl;
    cout << setw(40) << left << "Standard ticket Price: " << "Rs." << ticksPrice[0] << "/-"
<<endl;
    cout << setw(40) << left << "Premium Ticket Price: " << "Rs." << ticksPrice[1] << "/-"
<<endl;
    cout << setw(40) << left << "VIP Ticket Price: " << "Rs." << ticksPrice[2] << "/-" <<endl;
    cout <<endl;
}
```

```
void updatePriceInput(string type[],int ticksPrice[],string newPrice[]){        // Function
for admin to input new ticket prices
```

```
    for (int x = 0; x < 3; x++){
```

```
        cout << "Enter New " << type[x+1] << " Ticket Price: ";
        getline(cin>>ws,newPrice[x]);
        ticksPrice[x] = stoi(newPrice[x]);
    }
    cout << endl;
}
```

```cpp
void loadTicketPricesFile(string filename2,int ticksPrice[]){           // Function for storing ticket prices in a file

    fstream file;
    file.open(filename2,ios::out);
    for (int x = 0; x < 3; x++){

        file << ticksPrice[x];

        if (x < 2){
            file << ",";
        }

    }
    file.close();
}

void readTicketPricesFile(string filename2,int ticksPrice[]){       // Function for storing ticket prices stored in a file in their array

    fstream file;
    file.open(filename2,ios::in);
    string line;

    while (getline(file, line)){



        ticksPrice[0] = stoi(getField(line,1));
        ticksPrice[1] = stoi(getField(line,2));
        ticksPrice[2] = stoi(getField(line,3));
    }
    file.close();

}

// Function for admin to view sold tickets data

string soldTicketsData(int &ticks,int standticksSold[],string match[],string stands[],string type[],string &askMatchIndex,int typeSold[],int &quantityIndex){

    cout << endl;
    cout << "Sold Tickets Data >" <<endl<<endl;
    if (ticks == 0){

        soldTicketsDataOutput(standticksSold,match,stands,type,askMatchIndex,typeSold);
        return "No Tickets Have Been Sold Yet...";
    }
    else if (ticks > 0){

        soldTicketsDataOutput(standticksSold,match,stands,type,askMatchIndex,typeSold);
        return "This is The Current Sold Tickets Data...";
    }
}
```

```cpp
void soldTicketsDataOutput(int standticksSold[],string match[],string stands[],string
type[],string &askMatchIndex,int typeSold[]){    // Function for printing soldTicketsData
function's output
```

```cpp
    cout << setw(70) << left <<
"#############################################################################################
######" <<endl;
    cout << setw(50) << right << "Sold Tickets Data" <<endl;
    cout << setw(70) << left <<
"#############################################################################################
######" <<endl;
    cout << setw(20) << left << "Stands" << setw(15) << left << "Tickets Sold" << setw(20) <<
left << "Type" << setw(15) << left << "Tickets Sold" <<endl;
    cout << setw(70) << left <<
"#############################################################################################
######" <<endl;
    cout << setw(20) << left << stands[1] << setw(15) << left << standticksSold[0] << setw(20)
<< left << type[1] << setw(15) << left << typeSold[0] <<endl;
    cout << setw(20) << left << stands[2] << setw(15) << left << standticksSold[1] << setw(20)
<< left << type[2] << setw(15) << left << typeSold[1] <<endl;
    cout << setw(20) << left << stands[3] << setw(15) << left << standticksSold[2] << setw(20)
<< left << type[3] << setw(15) << left << typeSold[2] <<endl;
    cout << setw(20) << left << stands[4] << setw(15) << left << standticksSold[3] << setw(35)
<< left << " " <<endl;
    cout << setw(70) << left <<
"#############################################################################################
######" <<endl<<endl;
}
```

```cpp
void showFeedbacks(string feedback[],int &feedbackIndex){                   // Function for
admin to view customer feedbacks
```

```cpp
    cout <<endl;
    cout << "Customer Feedbacks >" <<endl<<endl;
    if (feedbackIndex == 0){
        cout << "There are no Feedbacks Yet...";
    }
    else if (feedbackIndex > 0){

        for (int x = 0; x < feedbackIndex; x++){
```

```cpp
            cout << "Feedback Of Customer " << x+1 << ": " << feedback[x];
            cout <<endl;
        }
    }
}

void showComplaint(string complaint[],int &complaintIndex){           // Function for customer
to view customer complaints
```

```cpp
    cout <<endl;
    cout << "Customer Complaints > " <<endl<<endl;
    if (complaintIndex == 0){
        cout << "There are no Complaints Yet...";
    }
    else if (complaintIndex > 0){

        for (int x = 0; x < complaintIndex; x++){
            cout << "Complaint Of Customer " << x+1 << ": " << complaint[x];
            cout <<endl;
        }
    }
}
```

```
string cafe(string cafeItems[],string cafePrice[],string newCafePrice[],int &cafeIndex,int
&items,string filename6){            // Function for admin to update cafe details
```

```
    while (true){

        cout << endl;
        cout << "Manage Cafe > " <<endl<<endl;
        showCafeOutput(cafeItems,cafePrice,cafeIndex,items);
        cout << "Do You Want To Update Cafe Details ('Yes' or 'No'): ";
        string update;
        getline(cin>>ws,update);
```

```
        if (update == "Yes" || update == "yes"){

            cout << endl;
            cafeOutput(cafeItems,cafePrice,newCafePrice,cafeIndex,items);
            loadCafeFile(filename6,cafePrice,items);
            cafeIndex++;
            break;
        }
        else if (update == "No" || update == "no"){
```

```
            cout <<endl;
            break;
        }
        else {
            wrongOption();
        }
    }
    return "Cafe Items have Been Managed...";
}
```

```
void cafeOutput(string cafeItems[],string cafePrice[],string newCafePrice[],int &cafeIndex,int
&items){     // function for printing cafe function's output

    for (int x = 0; x < items; x++){
```

```
        cout << "Enter New " << cafeItems[x] << " Price: ";
        cin >> newCafePrice[x];
        cafePrice[x] = newCafePrice[x];
    }
    cout << endl;
}
```

```
void loadCafeFile(string filename6,string cafePrice[],int &items){          // Function for
storing cafe data in a file
```

```
    fstream file;
    file.open(filename6,ios::out);
    for (int x = 0; x < items; x++){
```

```
        file << cafePrice[x];
        if (x < items-1){
```

```cpp
            file << ",";
        }
        file << endl;
    }
    file.close();
}
```

```cpp
void readCafeFile(string filename6,string cafePrice[],int &items){             // Function
for storing cafe data stored in a file into their respective arrays


    fstream file;
    file.open(filename6,ios::in);
    string line;
    int x = 0;
```

```cpp
    while (getline(file, line)){

```

```cpp
        cafePrice[x] = getField(line,1);
        x++;

    }

    file.close();
}
```

```cpp
string parkManage(string &parkingAsk,string &maintenance,int &park){        // Function for
admin to manage parking areas
```

```cpp
    cout << endl;
    cout << "Manage Parking Areas >" <<endl<<endl;
    parkMaintenance(parkingAsk,maintenance,park);
    return "Parking Areas Managed...";
}
```

```cpp
void parkMaintenance(string &parkingAsk,string &maintenance,int &park){     // Function for
asking admin if any parking area is under maintenance
```

```cpp
    while (true){

        cout << "Is there any parking area under maintenance ('Yes' or 'No'): ";
        getline(cin>>ws,parkingAsk);
        cout << endl;
        if (parkingAsk == "No"){

            break;
        }
        else if (parkingAsk == "Yes"){

            parkManageArea(maintenance,park);
            break;
        }
        else {
            wrongOption();
        }
    }
}
```

```cpp
void parkManageArea(string &maintenance,int &park){           // Function for asking admin
which parking area is under maintenance
```

```cpp
    while (true){

        cout << "Which area is under maintenance ('Underground' or 'Oustside'): ";
        getline(cin>>ws,maintenance);
        cout << endl;
        if (maintenance == "Underground" || maintenance == "Outside"){

            park++;
            break;
        }
        else {
            wrongOption();
        }
    }
}
```

9. Weakness in the Business Application

- The functions may or may not be single responsibility.
- The UI is presentable but more colouring could be added.

10. Future Directions

- Try to design all functions so that they perform only a single function.
- Add more colours to make the application more user friendly.
- Add more validations wherever necessary.

11. Conclusion

The course of Programming fundamentals has been quite exciting thus far. Learning C++ language from the respectable teachers was a fun journey. Facing new challenges nearly everyday helped me improve my problem solving skills and enhanced my patience level. The programming projects helped in polishing the skills in C++ language and I tried my level best to make my projects stand out. In future, I will try to work on the issues I faced in this journey and learn from the mistakes I made.