# TOUNAMENT MANAGEMENT SYSTEM



Session: 2021 – 2024

**Submitted by:**

Khadim Hussain      2021-CS-204

**Supervised by:**

Mam Maida Shahid

Department of Computer Science

**University of Engineering and Technology**

**Lahore, Pakistan**

# CONTENTS

[Project Description: 3](#_gjdgxs)

[Users of Application:](#_30j0zll) 3

[Functional Requirements:](#_1fob9te) 4

Wireframes of Application5

Data Structures: 11

Functions Prototypes: 11

Functions Working Flow: 13

Complete Code: 13

Test Cases for Project: 37

# PROJECT DESCRIPTION

My project is Tournament Management System. It can be used to manage a tournament. It can be used by both admin and user.TO run the program as admin, password is required. As an admin data of different teams can be added and managed in proper way. Data can be stored. Any unknown user can see that data and upcoming tournaments.

# USERS OF APPLICATION

There are two types of users of this management system application which are as follows:

* Admin
* User

# FUNCTIONAL REQUIREMENTS

**Admin:**

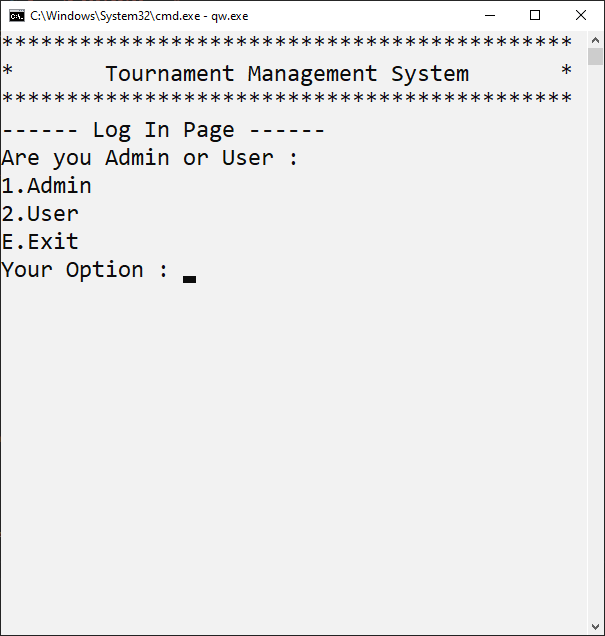
* Add Teams
* View All Teams
* Manage Teams
* Schedule Matches
* View Schedule
* Edit Point Table
* View Point Table
* Add Tournaments
* View Tournaments
* Change password
* Change profile

**Customers:**

* View All Teams
* View Schedule
* View Point Table
* View Tournaments
* Change profile

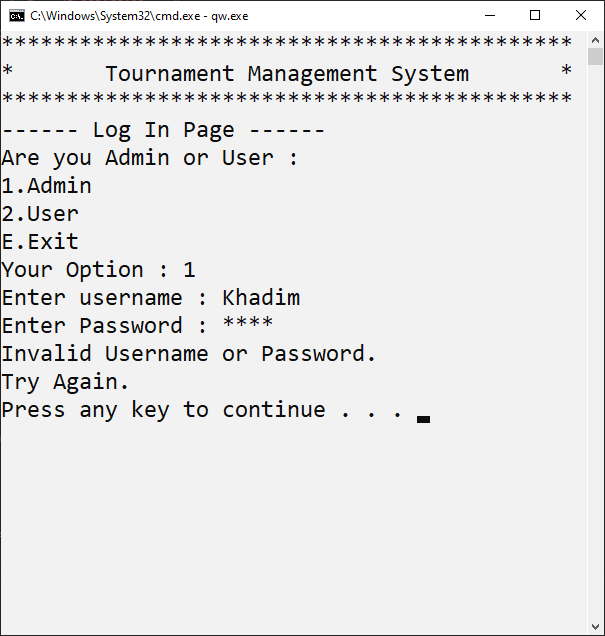
# WIRE FRAMES OF APPLICATION

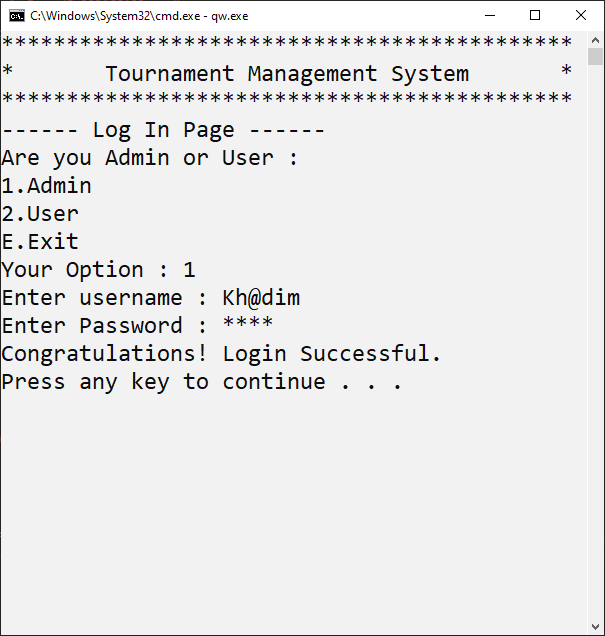
**Login** **Page:**



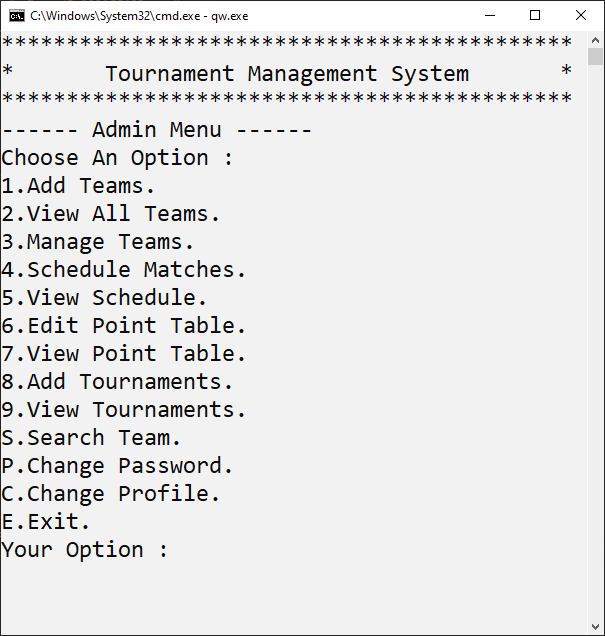
­­­­­­­­­­­­­­­­­­­

**Wrong Password: Correct Password:**

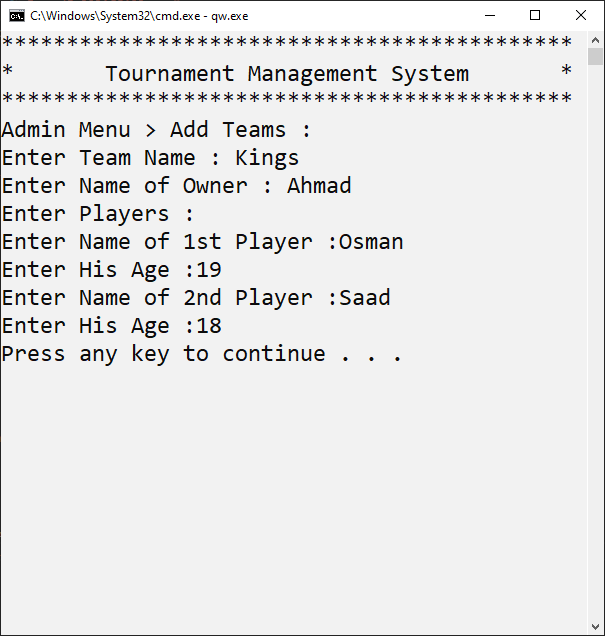




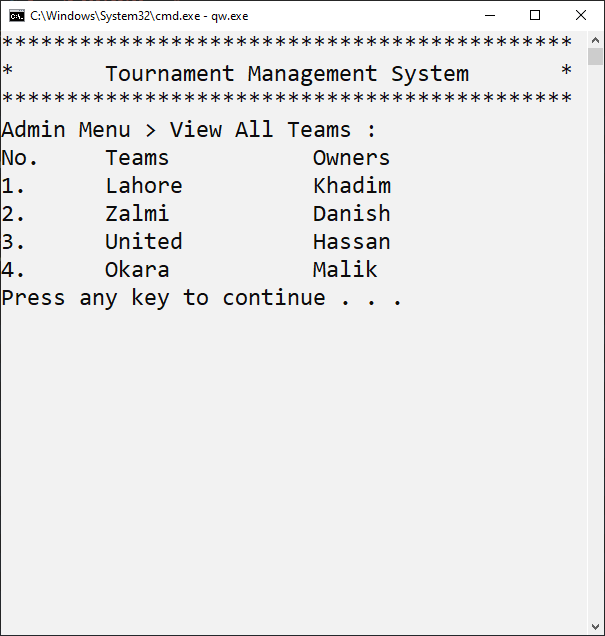
**1.Admin Menu:**

****

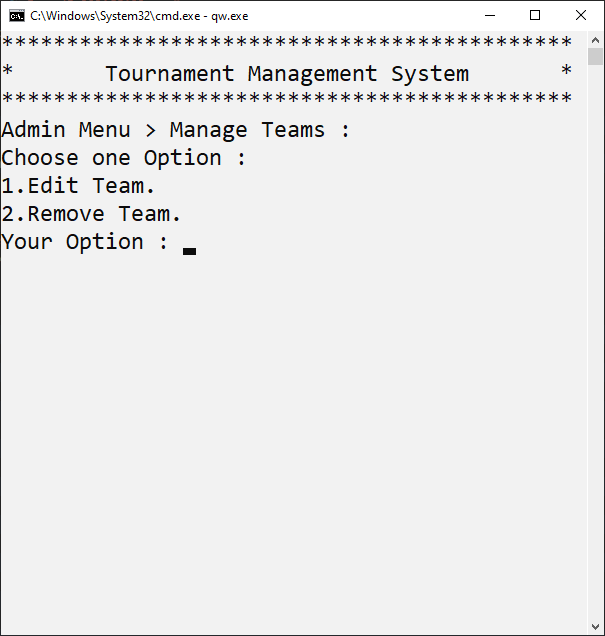
**1.1-Add Teams:**

****

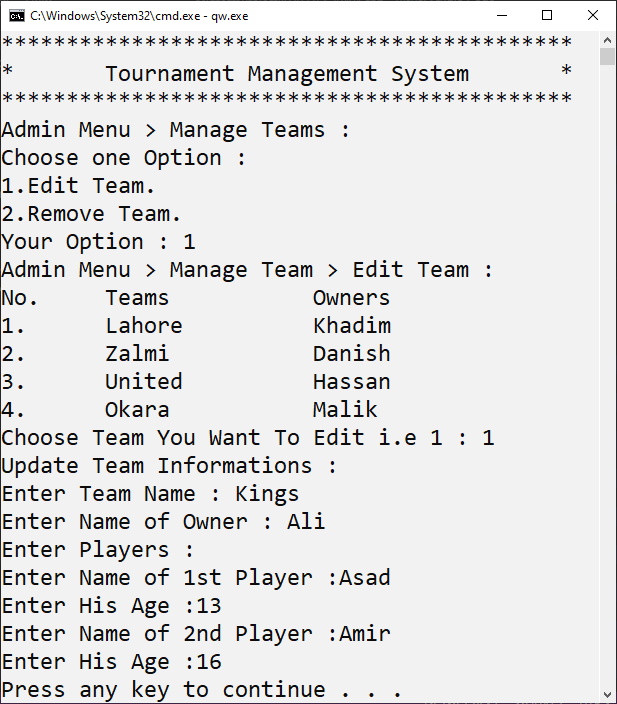
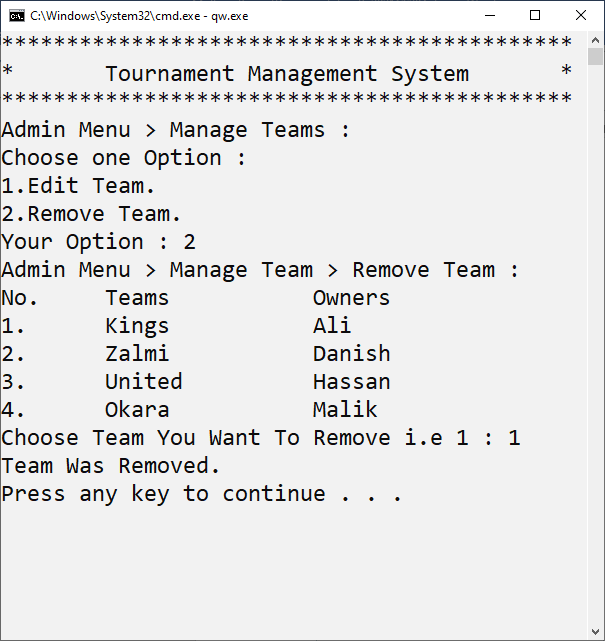
**1.2-View All Teams:**

****

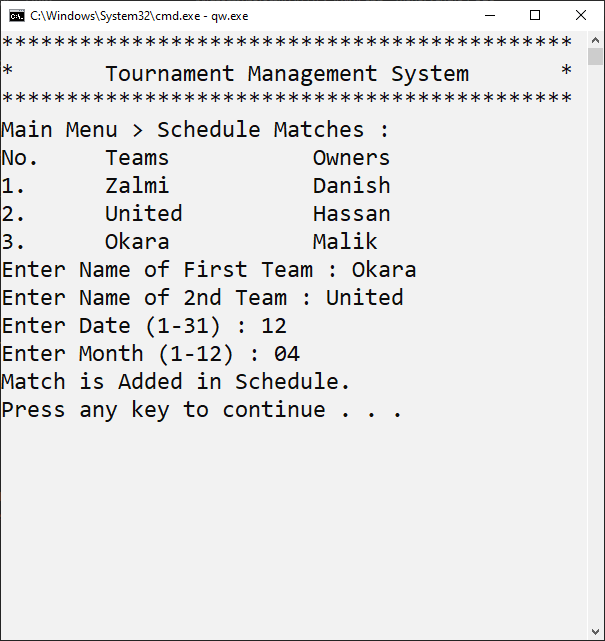
**1.3-Manage Teams:**

****

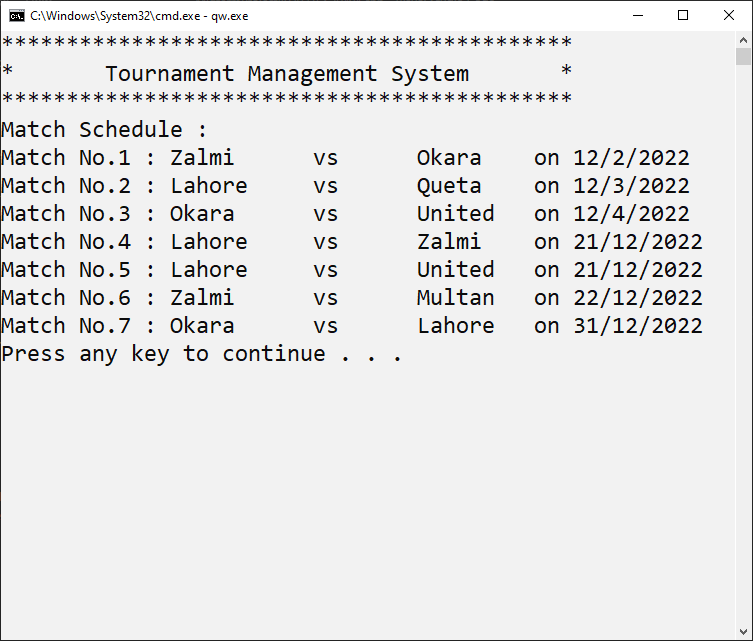
**Edit Team: Remove Teams:**

****

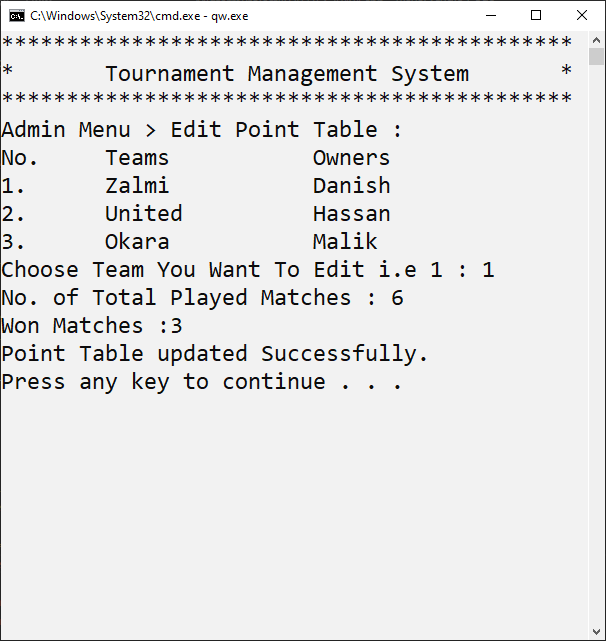
**1.4-Schedule Match:**

****

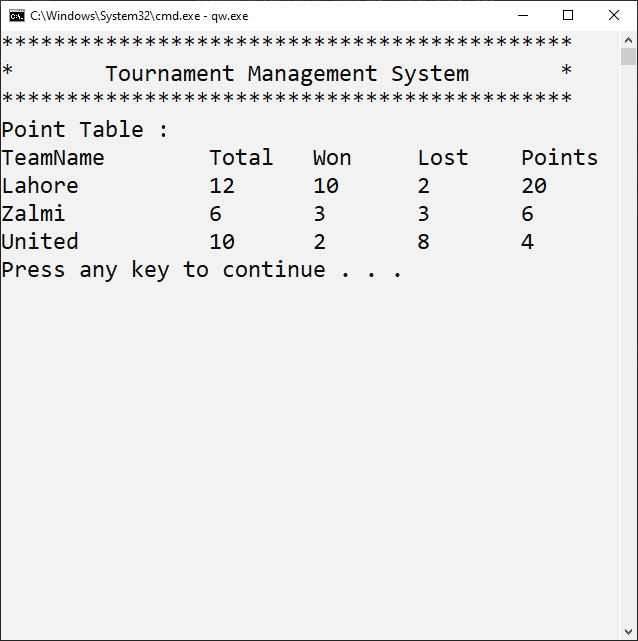
**1.5-View Schedule:**

****

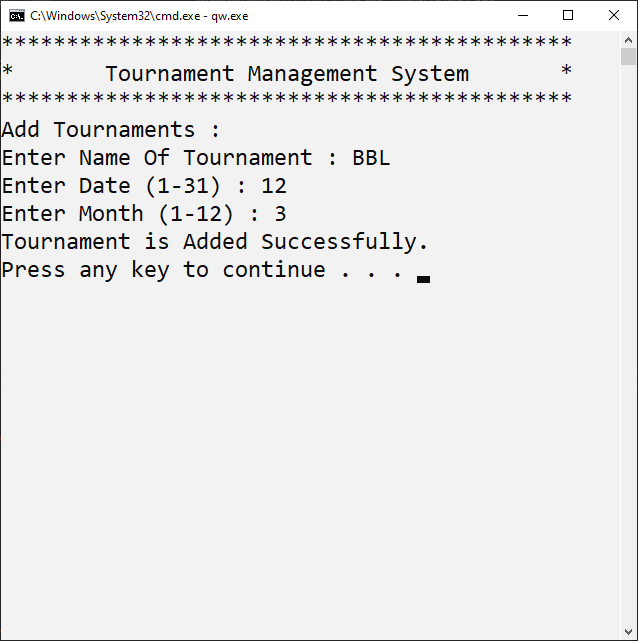
**1.6-Edit Point Table:**

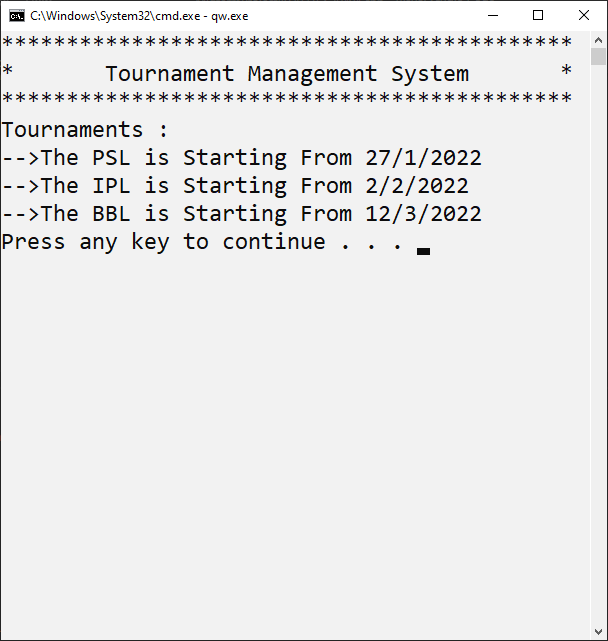
****

**1.7-View Point Table:**

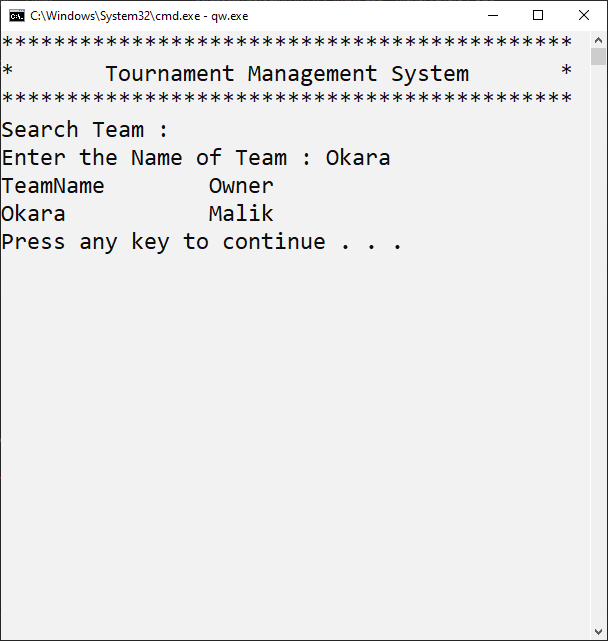
****

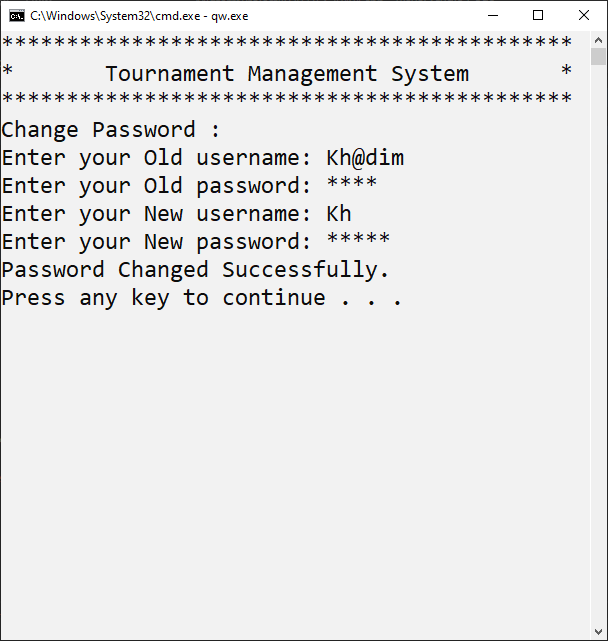
**1.8-Add Tournament:**

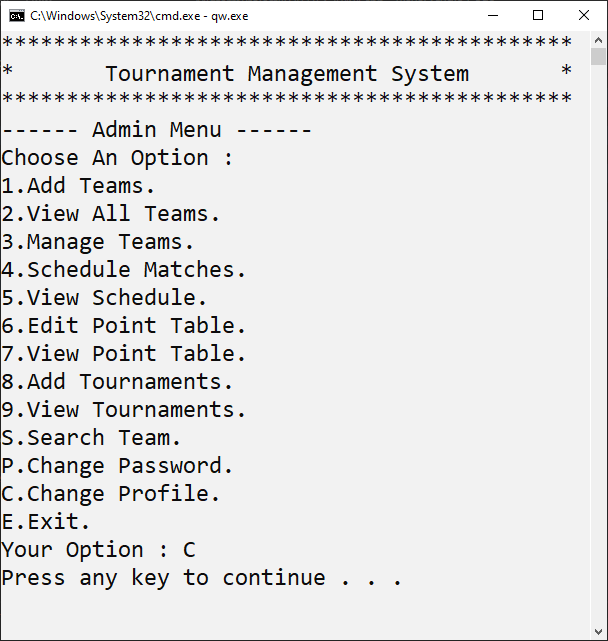
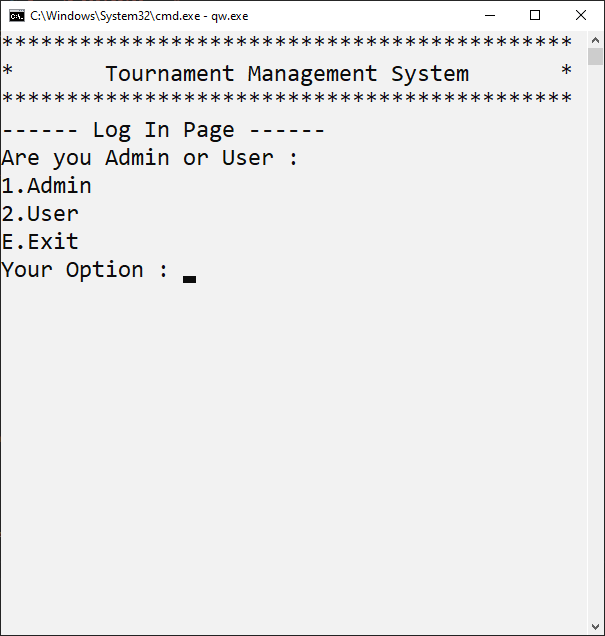
****

**1.9-View Tournaments:**

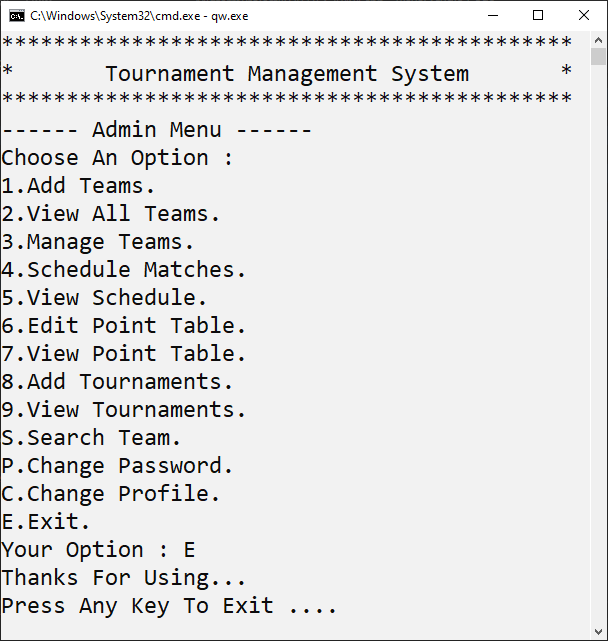
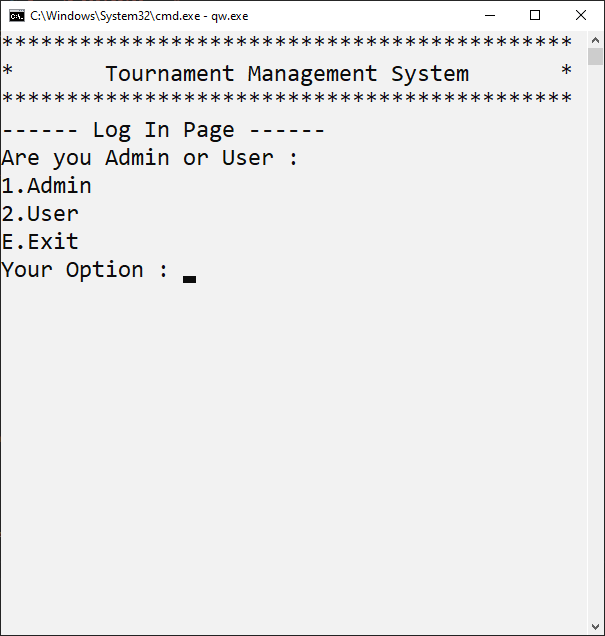
**1.10-Search Team:**

****

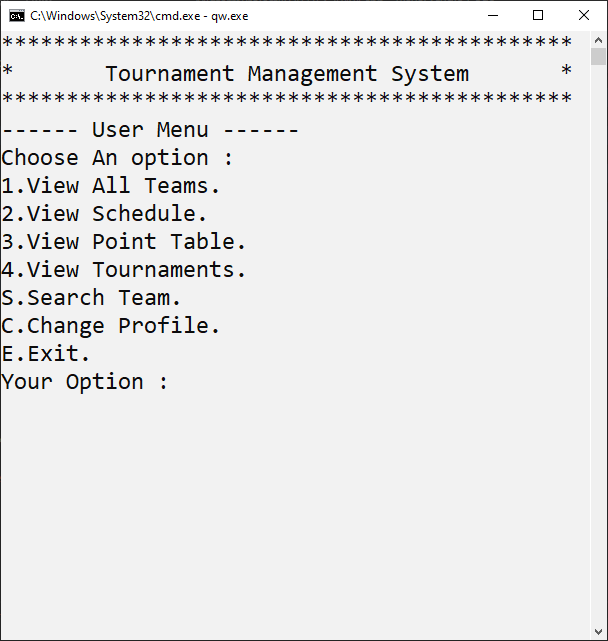
**1.11-Change Password:**

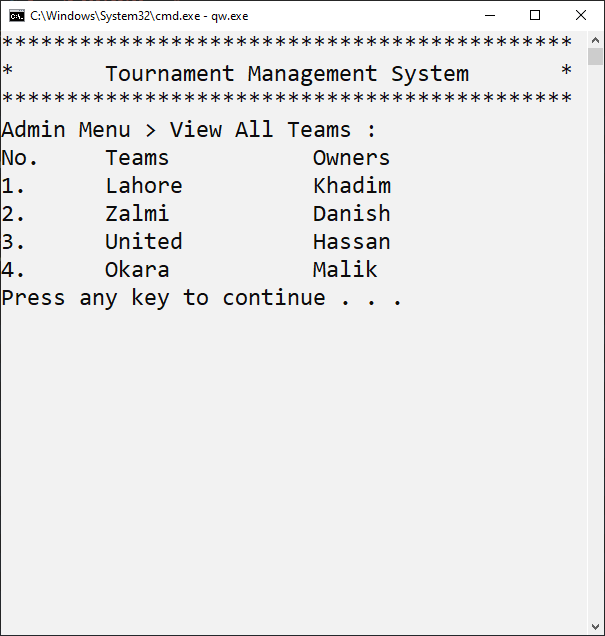
**1.12-Change Profile:**

**1.13-Exit:**

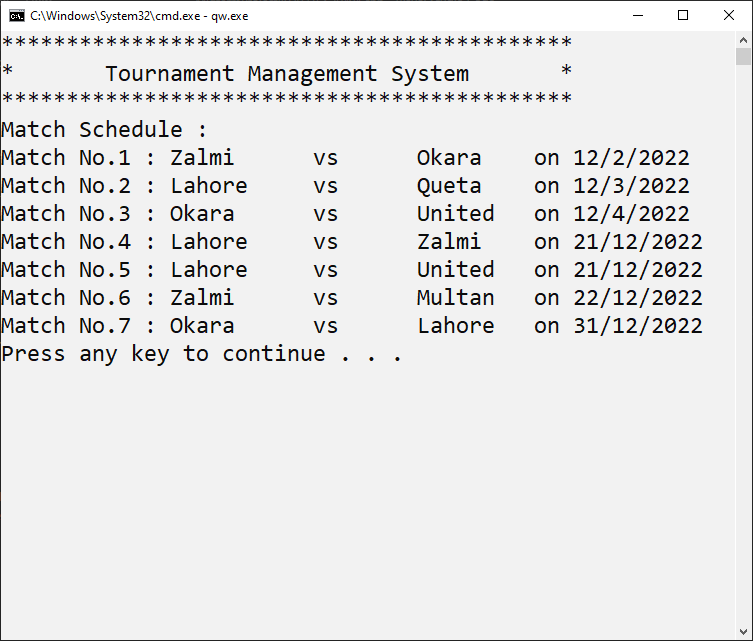
****

**2.User Menu:**

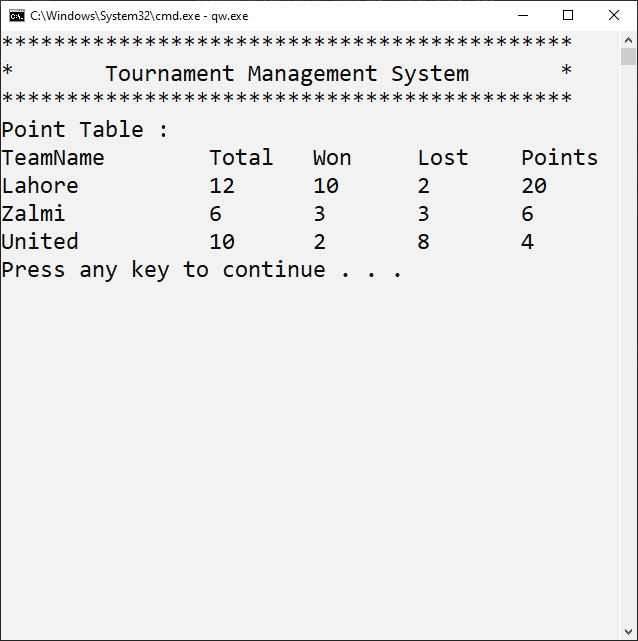
****

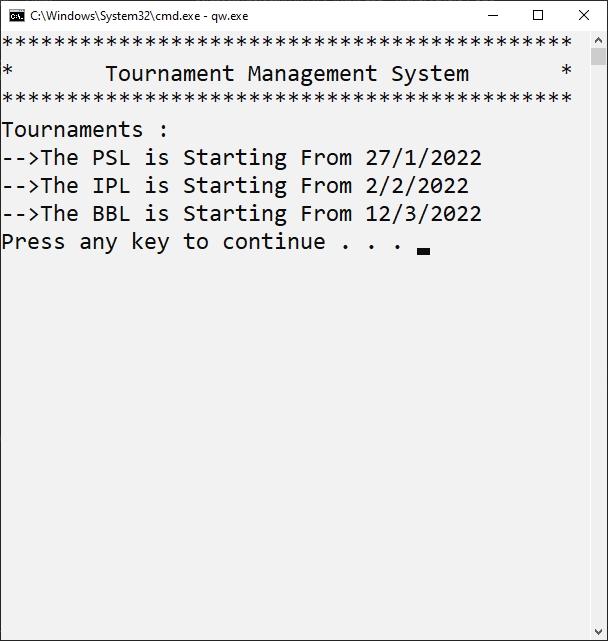
**2.1-View All Teams:**

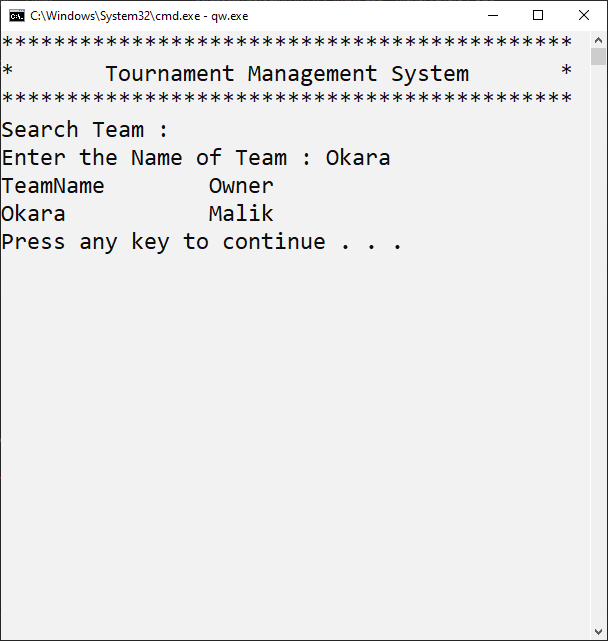
**2.2-View Schedule:**

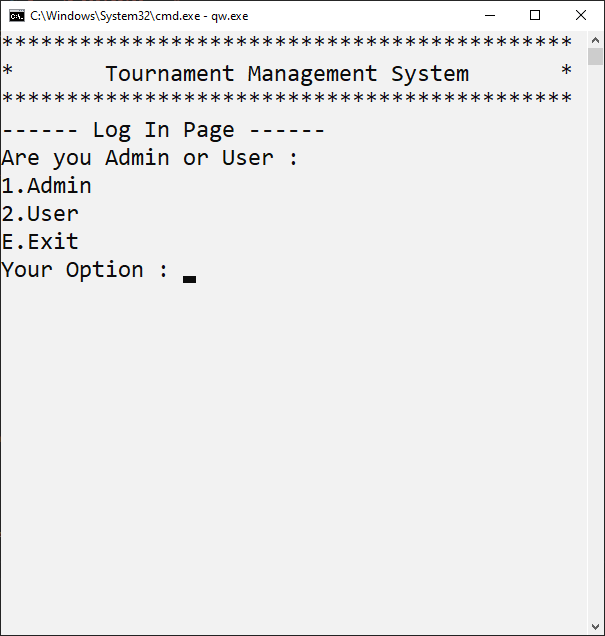
****

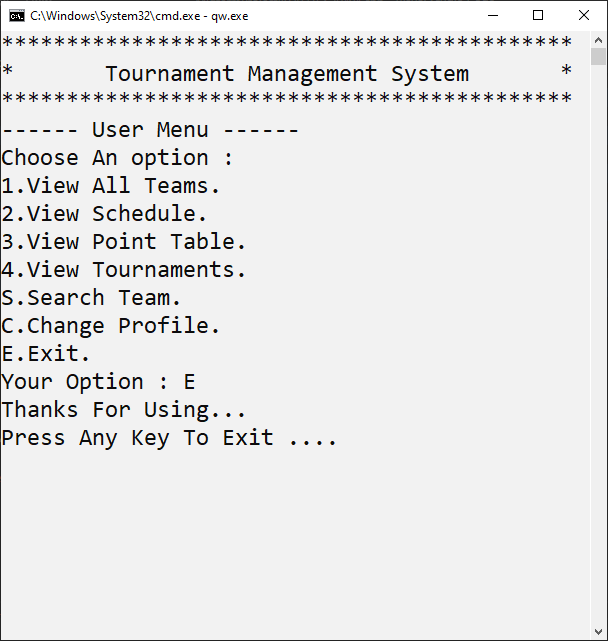
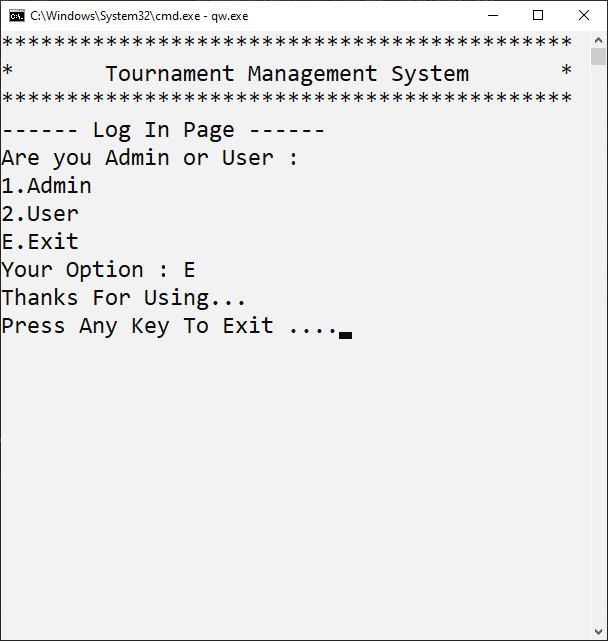
**2.3-View Point Table:**

****

**2.4-View Tournament:**

**2.5-Search Team:**

**2.6-Exit:**

**3.Exit:**

**DATA** **STRUCTURES**

**ARRAYS:**

const int MAX\_RECORD = 20;

string team\_name[MAX\_RECORD], owner\_name[MAX\_RECORD];

string team\_player1\_name[MAX\_RECORD], team\_player2\_name[MAX\_RECORD];

int team\_player1\_age[MAX\_RECORD], team\_player2\_age[MAX\_RECORD];

int team\_played\_matches[MAX\_RECORD], team\_won\_matches[MAX\_RECORD],

    team\_lost\_matches[MAX\_RECORD];

int team\_points[MAX\_RECORD];

string team\_name\_point[MAX\_RECORD];

string sorted\_team\_name[MAX\_RECORD], sorted\_owner\_name[MAX\_RECORD];

string sorted\_team\_player1\_name[MAX\_RECORD],

    sorted\_team\_player2\_name[MAX\_RECORD];

int sorted\_team\_player1\_age[MAX\_RECORD], sorted\_team\_player2\_age[MAX\_RECORD];

int sorted\_team\_played\_matches[MAX\_RECORD], sorted\_team\_won\_matches[MAX\_RECORD],

    sorted\_team\_lost\_matches[MAX\_RECORD];

int sorted\_team\_points[MAX\_RECORD];

string tounamentName[MAX\_RECORD];

int tournamentStratDate[MAX\_RECORD], tournamentStartMonth[MAX\_RECORD];

string smteam1name[MAX\_RECORD], smteam2name[MAX\_RECORD];

int smdate[MAX\_RECORD], smmonth[MAX\_RECORD], smtime[MAX\_RECORD];

# FUNCTIONS PROTOTYPES

void clearScreen();

void header();

char adminMenu();

char logIn();

char userMenu();

void admin();

void addTeam();

void addIntoArray(string teamName, string ownerName, string player1\_name,

                  int player1\_age, string player2\_name, int player2\_age);

void displayTeams();

void removeTeam();

void editTeam();

void manageTeams();

void scheduleMatches();

void matchSchedule();

void displaySchedule();

void editPointTable();

void displayPointTable();

void sorting();

void user();

void reArrangeTeams();

void storeTeamsInFile();

void loadTeamsFromFile();

void storePointsInFile();

void loadPointsFromFile();

void storeScheduleInFile();

void loadScheduleFromFile();

void storeData();

void loadData();

void scheduleInArray(string team1name, string team2name, int date, int month,

                     int time);

void Exit();

void storePassword();

void loadPassword();

void changePassword();

void teamMatch(int edit);

void tournaments();

void addTournaments();

void addTournamentsIntoArrays(string name, int date, int month);

void storeTournamentInFile();

void loadTournamentFromFile();

void searchTeam();

bool athentication();

void recomendation();

# FUNCTIONS WORKING FLOW

**COMPLETE** **CODE**

#include <conio.h>

#include <fstream>

#include <iostream>

#include <stdlib.h>

#include <windows.h>

using namespace std;

// Function ProtoTypes

void clearScreen();

void header();

char adminMenu();

char logIn();

char userMenu();

void admin();

void addTeam();

void addIntoArray(string teamName, string ownerName, string player1\_name,

int player1\_age, string player2\_name, int player2\_age);

void displayTeams();

void removeTeam();

void editTeam();

void manageTeams();

void scheduleMatches();

void matchSchedule();

void displaySchedule();

void editPointTable();

void displayPointTable();

void sorting();

void user();

void reArrangeTeams();

void storeTeamsInFile();

void loadTeamsFromFile();

void storePointsInFile();

void loadPointsFromFile();

void storeScheduleInFile();

void loadScheduleFromFile();

void storeData();

void loadData();

void scheduleInArray(string team1name, string team2name, int date, int month,

int time);

void Exit();

void storePassword();

void loadPassword();

void changePassword();

void teamMatch(int edit);

void tournaments();

void addTournaments();

void addTournamentsIntoArrays(string name, int date, int month);

void storeTournamentInFile();

void loadTournamentFromFile();

void searchTeam();

bool athentication();

string parsing(string line, int field);

// Data Structures

const int MAX\_RECORD = 20;

string team\_name[MAX\_RECORD], owner\_name[MAX\_RECORD];

string team\_player1\_name[MAX\_RECORD], team\_player2\_name[MAX\_RECORD];

int team\_player1\_age[MAX\_RECORD], team\_player2\_age[MAX\_RECORD];

int team\_played\_matches[MAX\_RECORD], team\_won\_matches[MAX\_RECORD],

team\_lost\_matches[MAX\_RECORD];

int team\_points[MAX\_RECORD];

string team\_name\_point[MAX\_RECORD];

string sorted\_team\_name[MAX\_RECORD], sorted\_owner\_name[MAX\_RECORD];

string sorted\_team\_player1\_name[MAX\_RECORD],

sorted\_team\_player2\_name[MAX\_RECORD];

int sorted\_team\_player1\_age[MAX\_RECORD], sorted\_team\_player2\_age[MAX\_RECORD];

int sorted\_team\_played\_matches[MAX\_RECORD], sorted\_team\_won\_matches[MAX\_RECORD],

sorted\_team\_lost\_matches[MAX\_RECORD];

int sorted\_team\_points[MAX\_RECORD];

int team\_count = 0;

string tounamentName[MAX\_RECORD];

int tournamentCount = 0;

int tournamentStartDate[MAX\_RECORD], tournamentStartMonth[MAX\_RECORD];

string smteam1name[MAX\_RECORD], smteam2name[MAX\_RECORD];

int smdate[MAX\_RECORD], smmonth[MAX\_RECORD], smtime[MAX\_RECORD];

int smcount = 0;

bool loading = true;

string userName ="Pakistan", password = "92";

// Functions

int main() {

if (loading) {

loadData();

loading = false;

}

system("cls");

char login = logIn();

if (login == '1') {

if (athentication()) {

clearScreen();

admin();

} else {

cout << "Invalid Username or Password.\nTry Again.\n";

clearScreen();

main();

}

} else if (login == '2') {

clearScreen();

user();

} else if (login == 'E') {

Exit();

} else {

cout << "Invalid Option.\nTry Again.\n";

clearScreen();

main();

}

}

void clearScreen() {

//\_\_\_\_clear screen\_\_\_\_\_

system("pause");

system("CLS");

}

void header() {

// Header

HANDLE l = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(l, 2);

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\* Tournament Management System \*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

HANDLE s = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(s, 7);

}

char adminMenu() {

// Admin Menu

header();

cout << "------ Admin Menu ------" << endl;

cout << "Choose An Option : " << endl;

cout << "1.Add Teams." << endl;

cout << "2.View All Teams." << endl;

cout << "3.Manage Teams." << endl;

cout << "4.Schedule Matches." << endl;

cout << "5.View Schedule." << endl;

cout << "6.Edit Point Table." << endl;

cout << "7.View Point Table." << endl;

cout << "8.Add Tournaments." << endl;

cout << "9.View Tournaments." << endl;

cout << "S.Search Team." << endl;

cout << "P.Change Password." << endl;

cout << "C.Change Profile." << endl;

cout << "E.Exit." << endl;

char option;

cout << "Your Option : ";

cin >> option;

return option;

}

char logIn() {

// Login Page

char login\_option;

header();

cout << "------ Log In Page ------" << endl;

cout << "Are you Admin or User : " << endl;

cout << "1.Admin" << endl;

cout << "2.User" << endl;

cout << "E.Exit" << endl;

cout << "Your Option : ";

cin >> login\_option;

return login\_option;

}

char userMenu() {

// User Menu

header();

cout << "------ User Menu ------" << endl;

cout << "Choose An option : " << endl;

cout << "1.View All Teams." << endl;

cout << "2.View Schedule." << endl;

cout << "3.View Point Table." << endl;

cout << "4.View Tournaments." << endl;

cout << "S.Search Team." << endl;

cout << "C.Change Profile." << endl;

cout << "E.Exit." << endl;

cout << "Your Option : ";

char option;

cin >> option;

return option;

}

void admin() {

// Admin Portion

system("cls");

while (true) {

char option = adminMenu();

if (option == 'E') {

Exit();

}

clearScreen();

header();

if (option == '1') {

// Add Team

addTeam();

} else if (option == '2') {

// Display Teams

if (team\_count >= 1) {

cout << "Admin Menu > View All Teams :" << endl;

displayTeams();

} else {

cout << "You should add at least 1 team to view it." << endl;

}

}

else if (option == '3') {

// Manage Teams

manageTeams();

} else if (option == '4') {

// Schedule Matches

scheduleMatches();

} else if (option == '5') {

// View Match Schedule

if (smcount >= 1) {

matchSchedule();

} else {

cout << "Currently No Schedule is Available." << endl;

}

} else if (option == '6') {

// Edit Point Table

editPointTable();

} else if (option == '7') {

// Display Point Table

if (team\_count >= 2) {

displayPointTable();

} else {

cout << "You Should Add At Least 2 Teams to view point Table." << endl;

}

} else if (option == 'C') {

// Login Page

system("cls");

main();

} else if (option == 'P') {

changePassword();

} else if (option == '8') {

addTournaments();

} else if (option == '9') {

tournaments();

} else if (option == 'S') {

searchTeam();

} else {

cout << "You Choosed Invalid Option ." << endl;

}

clearScreen();

}

}

void addTeam() {

// Add Team Information

string teamName = "\0", ownerName;

string player1\_name, player2\_name;

int player1\_age, player2\_age;

cout << "Admin Menu > Add Teams :" << endl;

cout << "Enter Team Name : ";

cin >> teamName;

cout << "Enter Name of Owner : ";

cin >> ownerName;

cout << "Enter Players : " << endl;

cout << "Enter Name of 1st Player :";

cin >> player1\_name;

cout << "Enter His Age :";

cin >> player1\_age;

cout << "Enter Name of 2nd Player :";

cin >> player2\_name;

cout << "Enter His Age :";

cin >> player2\_age;

// Add Team Information into Array

addIntoArray(teamName, ownerName, player1\_name, player1\_age, player2\_name,

player2\_age);

}

void addIntoArray(string teamName, string ownerName, string player1\_name,

int player1\_age, string player2\_name, int player2\_age) {

// Add Team Information into Array

team\_name[team\_count] = teamName;

team\_name\_point[team\_count] = teamName;

owner\_name[team\_count] = ownerName;

team\_player1\_name[team\_count] = player1\_name;

team\_player1\_age[team\_count] = player1\_age;

team\_player2\_name[team\_count] = player2\_name;

team\_player2\_age[team\_count] = player2\_age;

team\_count++;

}

void displayTeams() {

// Display Teams

cout << "No.\tTeams\t\tOwners" << endl;

for (int i = 0; i < team\_count; i++) {

cout << i + 1 << ".\t" << team\_name[i] << "\t\t" << owner\_name[i] << endl;

}

}

void removeTeam() {

// Remove Team

cout << "Admin Menu > Manage Team > Remove Team : " << endl;

displayTeams();

cout << "Choose Team You Want To Remove i.e 1 : ";

int edit;

cin >> edit;

int idx=-1;

if (edit <= team\_count) {

for(int i=0;i<team\_count;i++){

if(team\_name[edit-1]==team\_name\_point[i]){

idx=i;

}

}

team\_name\_point[idx] = "\0";

sorted\_team\_played\_matches[idx]=0;

sorted\_team\_won\_matches[idx]=0;

sorted\_team\_lost\_matches[idx]=0;

sorted\_team\_points[idx]=0;

team\_name[edit - 1] = "\0";

owner\_name[edit - 1] = "\0";

team\_player1\_name[edit - 1] = "\0";

team\_player1\_age[edit - 1] = 0;

team\_player2\_name[edit - 1] = "\0";

team\_player2\_age[edit - 1] = 0;

cout << "Team Was Removed." << endl;

reArrangeTeams();

} else {

cout << "Invalid Team." << endl;

}

}

void editTeam() {

// Edit Team

cout << "Admin Menu > Manage Team > Edit Team : " << endl;

displayTeams();

cout << "Choose Team You Want To Edit i.e 1 : ";

int edit;

cin >> edit;

if (edit <= team\_count) {

cout << "Update Team Informations : " << endl;

cout << "Enter Team Name : ";

cin >> team\_name[edit - 1];

team\_name\_point[edit - 1] = team\_name[edit - 1];

cout << "Enter Name of Owner : ";

cin >> owner\_name[edit - 1];

cout << "Enter Players : " << endl;

cout << "Enter Name of 1st Player :";

cin >> team\_player1\_name[edit - 1];

cout << "Enter His Age :";

cin >> team\_player1\_age[edit - 1];

cout << "Enter Name of 2nd Player :";

cin >> team\_player2\_name[edit - 1];

cout << "Enter His Age :";

cin >> team\_player2\_age[edit - 1];

} else {

cout << "Invalid Team." << endl;

}

}

void manageTeams() {

// Manage Teams

if (team\_count >= 1) {

cout << "Admin Menu > Manage Teams :" << endl;

cout << "Choose one Option : " << endl;

cout << "1.Edit Team." << endl;

cout << "2.Remove Team." << endl;

char op;

cout << "Your Option : ";

cin >> op;

if (op == '2') {

removeTeam();

} // End of op==2 ie Remove Team

else if (op == '1') {

editTeam();

} // End of op==1 ie Edit Team

else {

cout << "Invalid Option" << endl;

}

}

else {

cout << "You Should Add At Least 1 Team To Manage it." << endl;

}

}

void scheduleMatches() {

// Schedule Matches

string team1name, team2name;

int date, month, time;

bool check = false;

cout << "Main Menu > Schedule Matches : " << endl;

if (team\_count >= 2) {

displayTeams();

cout << "Enter Name of First Team : ";

cin >> team1name;

cout << "Enter Name of 2nd Team : ";

cin >> team2name;

if (team1name != team2name) {

for (int i = 0; i < team\_count; i++) {

if (team1name == team\_name[i]) {

for (int j = 0; j < team\_count; j++) {

if (team2name == team\_name[j]) {

while (true) {

cout << "Enter Date (1-31) : ";

cin >> date;

if (date < 1 || date > 31) {

cout << "Invalid Date.\nTry Again." << endl;

continue;

}

break;

}

while (true) {

cout << "Enter Month (1-12) : ";

cin >> month;

if (month < 1 || month > 12) {

cout << "Invalid Month.\nTry Again." << endl;

continue;

}

break;

}

time = date + (month \* 30);

cout<<"Match is Added in Schedule."<<endl;

scheduleInArray(team1name, team2name, date, month, time);

check = true;

break;

}

}

}

}

if (!check) {

cout << "Team Names Not Found in added Teams.\nTry Again." << endl;

cout << "Press Any Key to Try Again and 'e' to go to Admin Menu...";

char ch;

ch = getch();

if (ch == 'e')

admin();

system("cls");

header();

scheduleMatches();

}

} else {

cout << "A Team Cannot Play With Itself." << endl;

cout << "Press Any Key to Try Again and 'e' to go to Admin Menu...";

char ch;

ch = getch();

if (ch == 'e')

admin();

system("cls");

header();

scheduleMatches();

}

} else {

cout << "Please Enter At Least 2 Teams To Schedule Matches." << endl;

}

}

void matchSchedule() {

// Sort Matches with respect to Date

for (int i = 0; i < smcount; i++) {

for (int j = i + 1; j < smcount; j++) {

if (smtime[i] > smtime[j]) {

swap(smtime[i], smtime[j]);

swap(smdate[i], smdate[j]);

swap(smmonth[i], smmonth[j]);

swap(smteam1name[i], smteam1name[j]);

swap(smteam2name[i], smteam2name[j]);

}

}

}

displaySchedule();

}

void displaySchedule() {

// Display Schedule

cout << "Match Schedule : " << endl;

for (int i = 0; i < smcount; i++) {

cout << "Match No." << i + 1 << " : " << smteam1name[i] << "\tvs\t"

<< smteam2name[i] << " \t on " << smdate[i] << "/" << smmonth[i]

<< "/2022" << endl;

}

}

void editPointTable() {

// Edit Point Table

cout << "Admin Menu > Edit Point Table : " << endl;

if (team\_count >= 2) {

displayTeams();

cout << "Choose Team You Want To Edit i.e 1 : ";

int edit;

cin >> edit;

if (edit <= team\_count) {

teamMatch(edit);

} else {

cout << "Invalid Team." << endl;

}

} else {

cout << "You Should Add Two Teams To Edit Point Table." << endl;

}

}

void displayPointTable() {

// Display Point Table (Sorted)

cout << "Point Table : " << endl;

cout << "TeamName\tTotal\tWon\tLost\tPoints" << endl;

sorting();

for (int i = 0; i < team\_count; i++) {

cout << team\_name\_point[i] << "\t\t" << sorted\_team\_played\_matches[i]

<< "\t" << sorted\_team\_won\_matches[i] << "\t"

<< sorted\_team\_lost\_matches[i] << "\t" << sorted\_team\_points[i]

<< endl;

}

}

void sorting() {

// Sort Point Table

for (int i = 0; i < team\_count; i++) {

for (int j = i + 1; j < team\_count; j++) {

if (sorted\_team\_points[i] < sorted\_team\_points[j]) {

swap(sorted\_team\_points[i], sorted\_team\_points[j]);

swap(sorted\_team\_played\_matches[i], sorted\_team\_played\_matches[j]);

swap(sorted\_team\_won\_matches[i], sorted\_team\_won\_matches[j]);

swap(sorted\_team\_lost\_matches[i], sorted\_team\_lost\_matches[j]);

swap(team\_name\_point[i], team\_name\_point[j]);

}

}

}

}

void user() {

// User Portion

while (true) {

system("cls");

char option1 = userMenu();

if (option1 == 'E') {

Exit();

}

clearScreen();

header();

if (option1 == '1') {

// View All Teams

if (team\_count >= 1) {

cout << "User Menu > View All Teams :" << endl;

displayTeams();

} else {

cout << "Not Enough Data To Show." << endl;

}

} else if (option1 == '2') {

// View Match Schedule

if (smcount >= 1) {

matchSchedule();

} else {

cout << "Currently No Schedule is Available." << endl;

}

} else if (option1 == '3') {

// Display Point Table

if (team\_count < 2) {

cout << "Not Enough Data To Show." << endl;

} else {

displayPointTable();

}

} else if (option1 == 'C') {

// Change Profile

system("cls");

main();

} else if (option1 == '4') {

tournaments();

} else if (option1 == 'S') {

searchTeam();

} else {

cout << "Invalid Option." << endl;

}

clearScreen();

}

}

void reArrangeTeams() {

// ReArrange Teams After team is removed

for (int i = 0; i < team\_count; i++) {

for (int j = i + 1; j < team\_count; j++) {

if (team\_name[i] == "\0") {

swap(team\_name[i], team\_name[j]);

swap(team\_name\_point[i], team\_name\_point[j]);

swap(owner\_name[i], owner\_name[j]);

swap(team\_player1\_name[i], team\_player1\_name[j]);

swap(team\_player1\_age[i], team\_player1\_age[j]);

swap(team\_player2\_name[i], team\_player2\_name[j]);

swap(team\_player2\_age[i], team\_player2\_age[j]);

}

}

}

for (int i = 0; i < team\_count; i++) {

for (int j = i + 1; j < team\_count; j++) {

if (team\_name\_point[i] == "\0") {

swap(team\_name\_point[i], team\_name\_point[j]);

swap(sorted\_team\_played\_matches[i], sorted\_team\_played\_matches[j]);

swap(sorted\_team\_won\_matches[i], sorted\_team\_won\_matches[j]);

swap(sorted\_team\_lost\_matches[i], sorted\_team\_lost\_matches[j]);

swap(sorted\_team\_points[i], sorted\_team\_points[j]);

}

}

}

team\_count--;

}

void storeTeamsInFile() {

// Store Teams Data in txt File

fstream file;

file.open("TeamInformation.txt", ios::out);

for (int i = 0; i < team\_count; i++) {

file << team\_name[i] << "," << owner\_name[i] << "," << team\_player1\_name[i]

<< "," << team\_player1\_age[i] << "," << team\_player2\_name[i] << ","

<< team\_player2\_age[i] << endl;

}

file.close();

}

void loadTeamsFromFile() {

// Load Teams Data in txt File

fstream file;

string teamName = "\0", ownerName, name;

string player1\_name, player2\_name;

int player1\_age, player2\_age;

int count = 0;

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

while (!file.eof()) {

getline(file, name);

count++;

}

file.close();

string line;

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

for (int i = 0; i < count - 1; i++) {

getline(file,line);

teamName=parsing(line,1);

ownerName=parsing(line,2);

player1\_name=parsing(line,3);

player1\_age=stoi(parsing(line,4));

player2\_name=parsing(line,5);

player2\_age=stoi(parsing(line,6));

addIntoArray(teamName, ownerName, player1\_name, player1\_age, player2\_name,

player2\_age);

}

file.close();

}

void storePointsInFile() {

// Store Teams Points Data in txt File

fstream file;

file.open("TeamPoints.txt", ios::out);

for (int i = 0; i < team\_count; i++) {

file << team\_name\_point[i] << "," << sorted\_team\_played\_matches[i] << ","

<< sorted\_team\_won\_matches[i] << "," << sorted\_team\_lost\_matches[i]

<< "," << sorted\_team\_points[i] << endl;

}

file.close();

}

void loadPointsFromFile() {

// Load Teams Points Data in txt File

fstream file;

int count = 0;

string name;

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

while (!file.eof()) {

getline(file, name);

count++;

}

file.close();

string line;

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

for (int i = 0; i < count - 1; i++) {

getline(file,line);

team\_name\_point[i]=parsing(line,1);

sorted\_team\_played\_matches[i]=stoi(parsing(line,2));

sorted\_team\_won\_matches[i]=stoi(parsing(line,3));

sorted\_team\_lost\_matches[i]=stoi(parsing(line,4));

sorted\_team\_points[i]=stoi(parsing(line,5));

}

file.close();

}

void scheduleInArray(string team1name, string team2name, int date, int month,

int time) {

// Add Schedule in Array

smteam1name[smcount] = team1name;

smteam2name[smcount] = team2name;

smdate[smcount] = date;

smmonth[smcount] = month;

smtime[smcount] = time;

smcount++;

}

void storeScheduleInFile() {

// Store Teams Schedule Data in txt File

fstream file;

file.open("TeamSchedule.txt", ios::out);

for (int i = 0; i < smcount; i++) {

file << smteam1name[i] << "," << smteam2name[i] << "," << smdate[i] << ","

<< smmonth[i] << "," << smtime[i] << endl;

}

file.close();

}

void loadScheduleFromFile() {

// Load Teams Schedule Data in txt File

fstream file;

string team1name, team2name, name;

int date, month, time;

int count = 0;

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

while (!file.eof()) {

getline(file, name);

count++;

}

file.close();

string line;

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

for (int i = 0; i < count - 1; i++) {

getline(file,line);

team1name=parsing(line,1);

team2name=parsing(line,2);

date=stoi(parsing(line,3));

month=stoi(parsing(line,4));

time=stoi(parsing(line,5));

scheduleInArray(team1name, team2name, date, month, time);

}

file.close();

}

void storeData() {

// Store All Data in txt files

storePassword();

storeScheduleInFile();

storePointsInFile();

storeTeamsInFile();

storeTournamentInFile();

}

void loadData() {

// Load All Data from txt files

loadTeamsFromFile();

loadPassword();

loadScheduleFromFile();

loadPointsFromFile();

loadTournamentFromFile();

}

void Exit() {

// Exit Program

cout << "Thanks For Using... \n";

storeData();

cout << "Press Any Key To Exit ....";

getch();

system("cls");

exit(1);

}

void changePassword() {

bool find = false;

cout << "Change Password : " << endl;

string username, pasword;

cout << "Enter your Old username: ";

cin >> username;

cout << "Enter your Old password: ";

cin >> pasword;

if (username == userName && pasword == password) {

cout << "Enter your New username: ";

cin >> username;

cout << "Enter your New password: ";

cin >> pasword;

for(int i=0;pasword[i]!='\0';i++){

if(pasword[i]==',')

{

find = true;

}

} for(int i=0;username[i]!='\0';i++){

if(username[i]==',')

{

find = true;

}

}

if(find){

cout<<"You Cannot Add Comma in Your Username or Password."<<endl;

}else{

userName =username ; password =pasword;

cout << "Password Changed Successfully.\n";

}

} else {

cout << "Invalid Username or Password.\n";

}

}

void loadPassword() {

fstream file;

string user, pass;

string line;

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

getline(file,line);

user=parsing(line,1);

pass=parsing(line,2);

file.close();

if (user != "\0") {

userName = user;

password = pass;

}

}

void storePassword() {

fstream file;

file.open("Password.txt", ios::out);

file << userName << "," << password << endl;

file.close();

}

void teamMatch(int edit) {

int total, won;

cout << "No. of Total Played Matches : ";

cin >> total;

cout << "Won Matches :";

cin >> won;

if (won <= total) {

team\_won\_matches[edit - 1] = won;

team\_played\_matches[edit - 1] = total;

team\_lost\_matches[edit - 1] =

team\_played\_matches[edit - 1] - team\_won\_matches[edit - 1];

team\_points[edit - 1] = 2 \* team\_won\_matches[edit - 1];

int idx = 0;

for (int i = 0; i < team\_count; i++) {

if (team\_name[edit - 1] == team\_name\_point[i]) {

idx = i;

}

}

sorted\_team\_won\_matches[idx] = team\_won\_matches[edit - 1];

sorted\_team\_played\_matches[idx] = team\_played\_matches[edit - 1];

sorted\_team\_lost\_matches[idx] = team\_lost\_matches[edit - 1];

sorted\_team\_points[idx] = team\_points[edit - 1];

cout<<"Point Table updated Successfully."<<endl;

} else {

cout << "Won Matches Cannot exceed Total Matches.\n";

cout << "Enter Carrect Information.\n";

teamMatch(edit);

}

}

void tournaments() {

cout << "Tournaments : " << endl;

if (tournamentCount == 0) {

cout << "Due To Covid-19 Tounaments are Postponed." << endl;

cout << "For New Updates Keep Visiting." << endl;

} else {

for (int i = 0; i < tournamentCount; i++) {

cout << "-->The " << tounamentName[i] << " is Starting From "

<< tournamentStartDate[i] << "/" << tournamentStartMonth[i]

<< "/2022" << endl;

}

}

}

void addTournaments() {

cout << "Add Tournaments : " << endl;

string name;

int date, month;

cout << "Enter Name Of Tournament : ";

cin >> name;

while (true) {

cout << "Enter Date (1-31) : ";

cin >> date;

if (date < 1 || date > 31) {

cout << "Invalid Date.\nTry Again." << endl;

continue;

}

break;

}

while (true) {

cout << "Enter Month (1-12) : ";

cin >> month;

if (month < 1 || month > 12) {

cout << "Invalid Month.\nTry Again." << endl;

continue;

}

break;

}

cout<<"Tournament is Added Successfully."<<endl;

addTournamentsIntoArrays(name, date, month);

}

void addTournamentsIntoArrays(string name, int date, int month) {

tounamentName[tournamentCount] = name;

tournamentStartDate[tournamentCount] = date;

tournamentStartMonth[tournamentCount] = month;

tournamentCount++;

}

void storeTournamentInFile() {

fstream file;

file.open("Tournament.txt", ios::out);

for (int i = 0; i < tournamentCount; i++) {

file << tounamentName[i] << "," << tournamentStartDate[i] << ","

<< tournamentStartMonth[i] << endl;

}

file.close();

}

void loadTournamentFromFile() {

fstream file;

int count = 0;

string name;

int date, month;

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

while (!file.eof()) {

getline(file, name);

count++;

}

file.close();

string line;

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

for (int i = 0; i < count - 1; i++) {

getline(file,line);

name=parsing(line,1);

date=stoi(parsing(line,2));

month=stoi(parsing(line,3));

addTournamentsIntoArrays(name, date, month);

}

file.close();

}

void searchTeam() {

string name;

bool b = true;

cout << "Search Team : " << endl;

cout << "Enter the Name of Team : ";

cin >> name;

for (int i = 0; i < team\_count; i++) {

if (team\_name[i] == name) {

cout << "TeamName\tOwner\n";

cout << team\_name[i] << "\t\t" << owner\_name[i] << endl;

b = false;

}

}

if (b) {

cout << "No Team found With This Name. " << endl;

}

}

bool athentication() {

string username, password1;

cout << "Enter username : ";

cin >> username;

cout << "Enter Password : ";

cin>>password1;

if (userName == username && password == password1) {

cout<<"Congratulations! Login Successful."<<endl;

return true;

}

return false;

}

string parsing(string line, int field)

{

int comma = 1;

string item;

for (int i = 0; i < line.length(); i++)

{

if (line[i] == ',')

{

comma++;

}

else if (field == comma)

{

item = item + line[i];

}

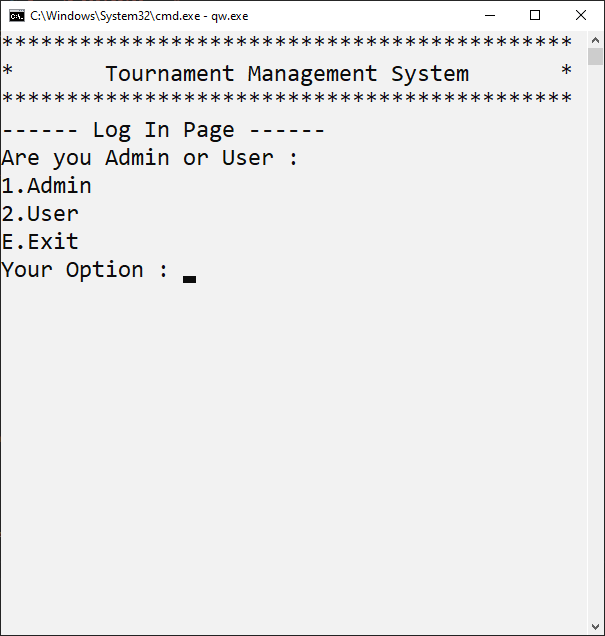
}

return item;

}

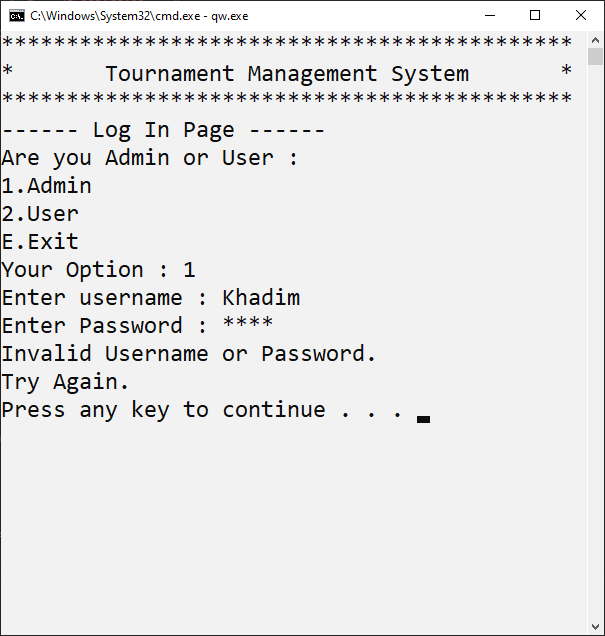
**Test Cases for Project**

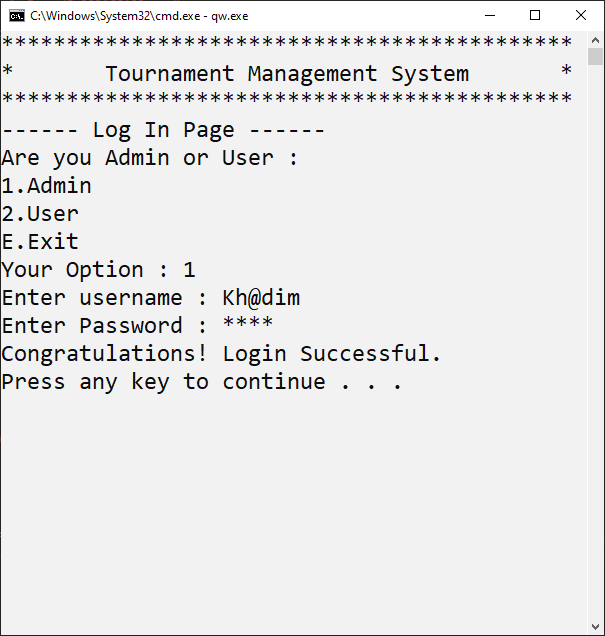
**Login** **Page:**



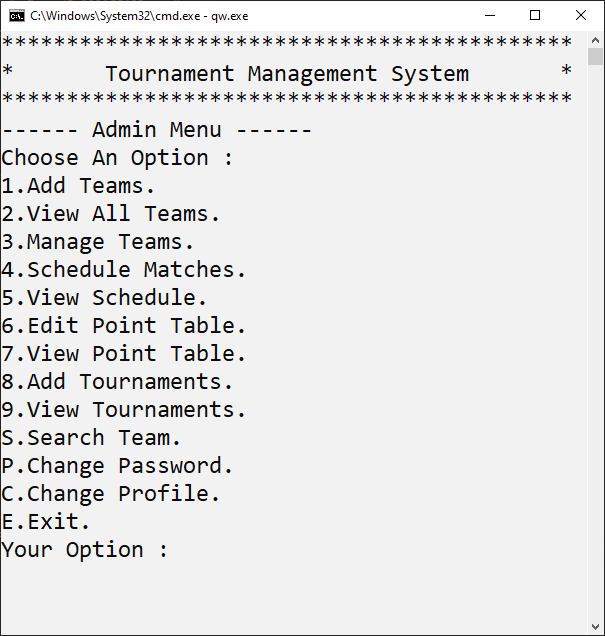
­­­­­­­­­­­­­­­­­­­

**Wrong Password: Correct Password:**

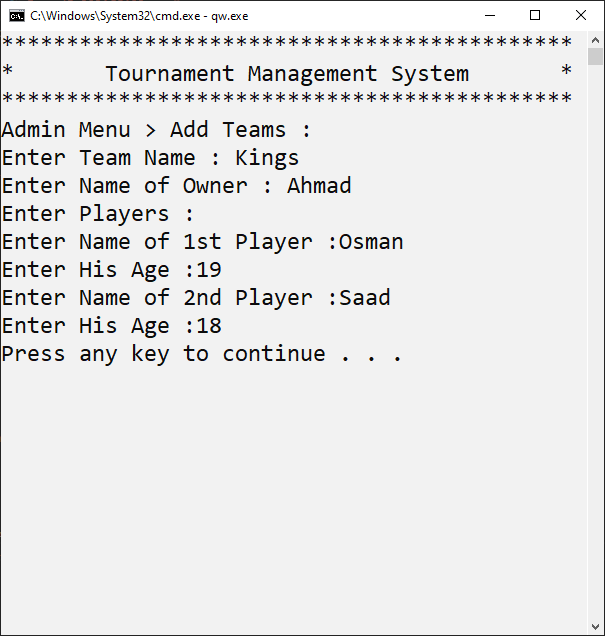




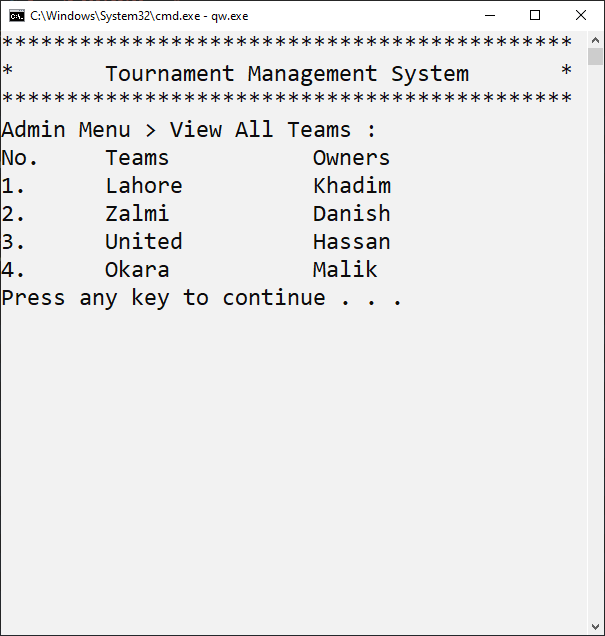
**1.Admin Menu:**

****

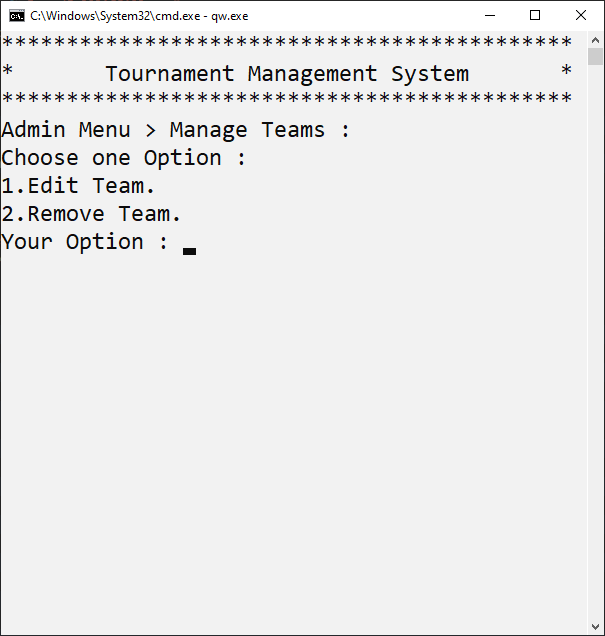
**1.1-Add Teams:**

****

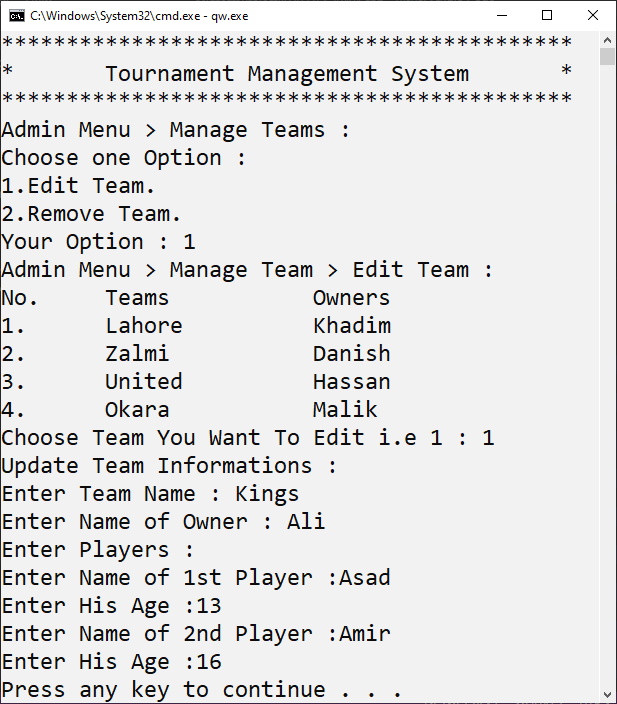
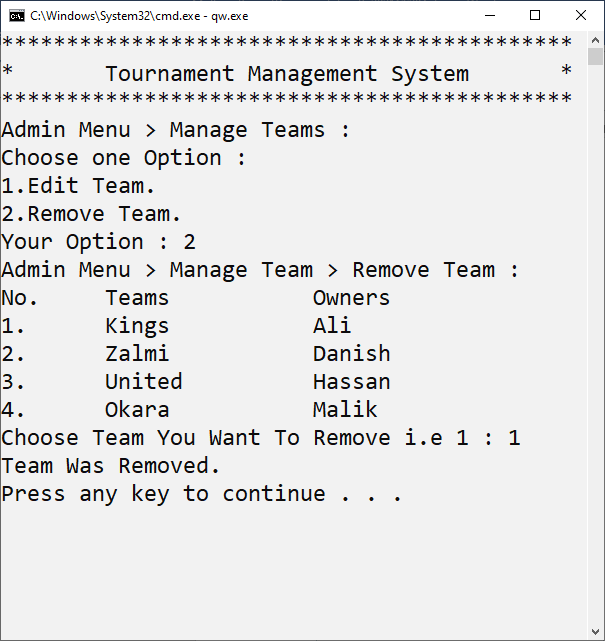
**1.2-View All Teams:**

****

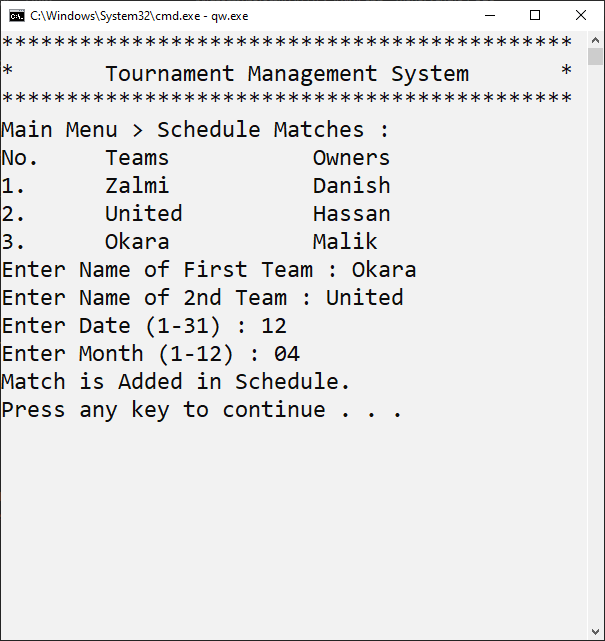
**1.3-Manage Teams:**

****

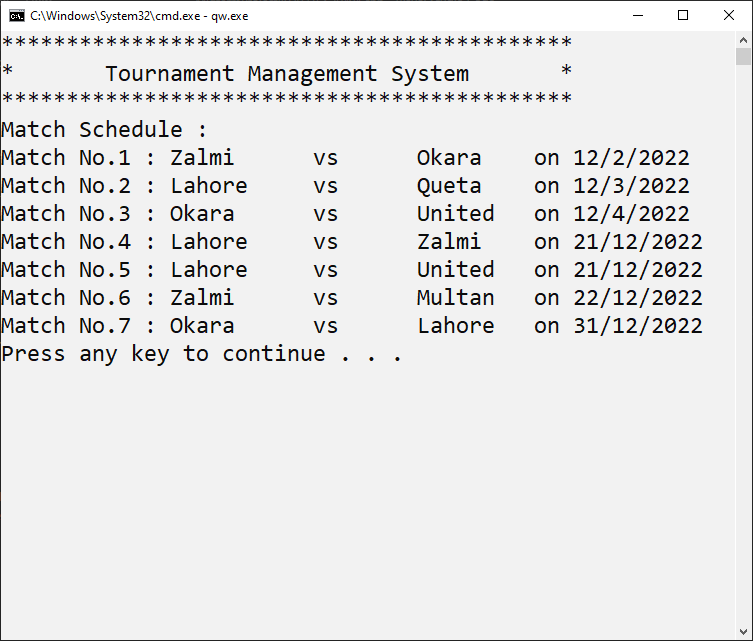
**Edit Team: Remove Teams:**

****

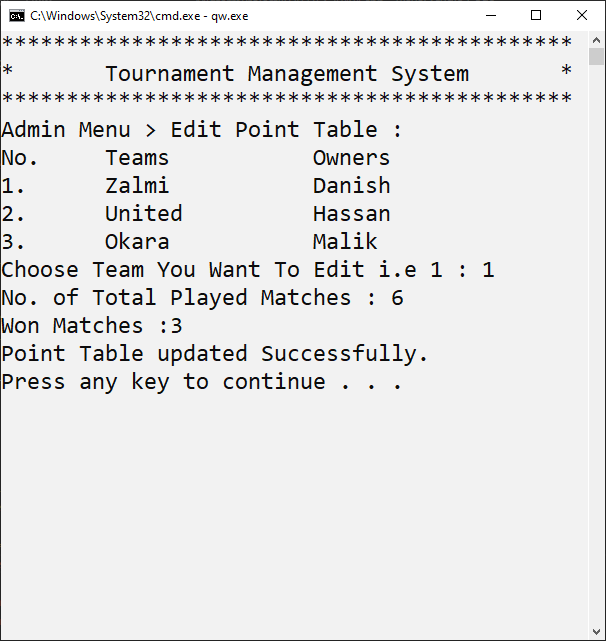
**1.4-Schedule Match:**

****

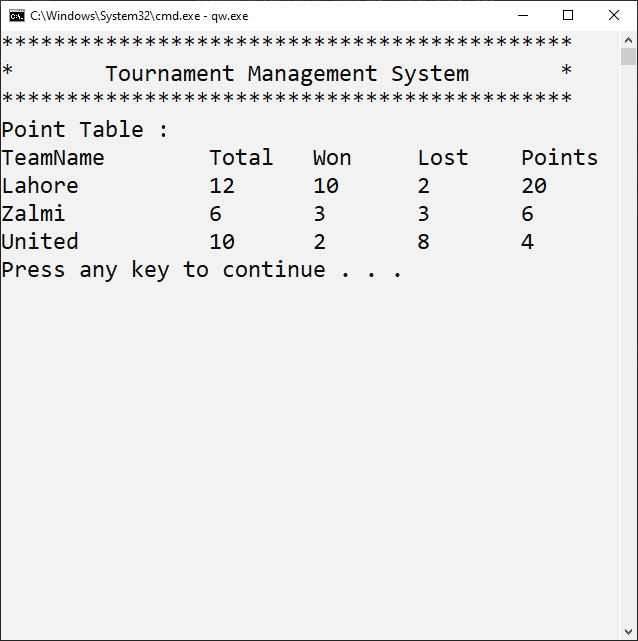
**1.5-View Schedule:**

****

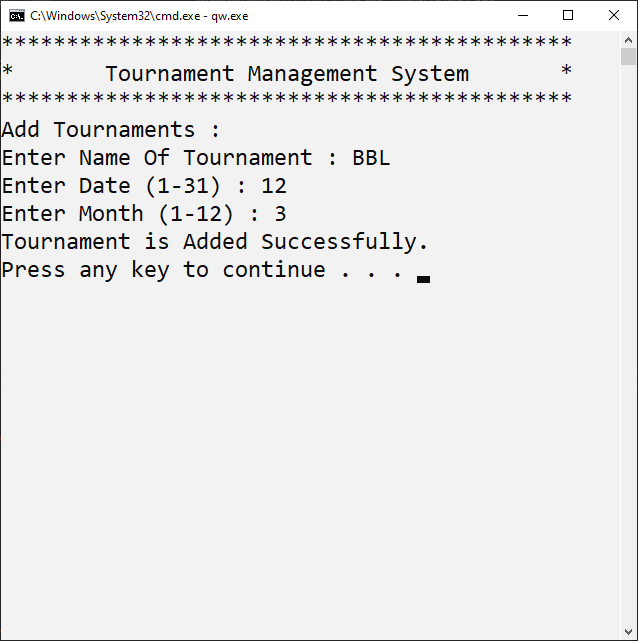
**1.6-Edit Point Table:**

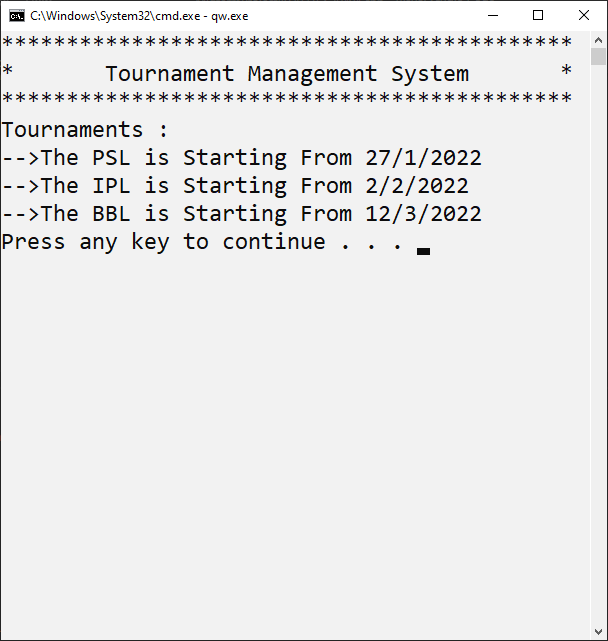
****

**1.7-View Point Table:**

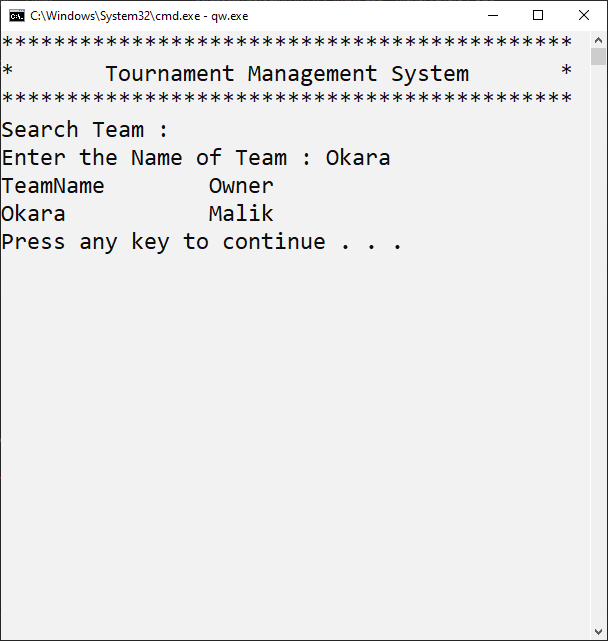
****

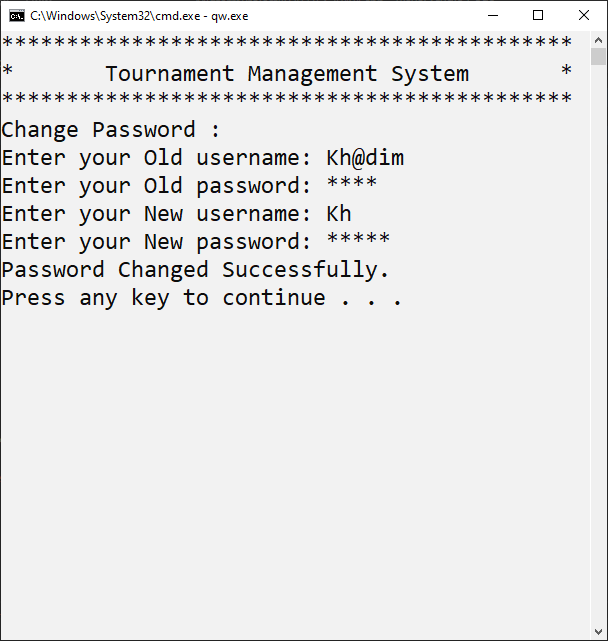
**1.8-Add Tournament:**

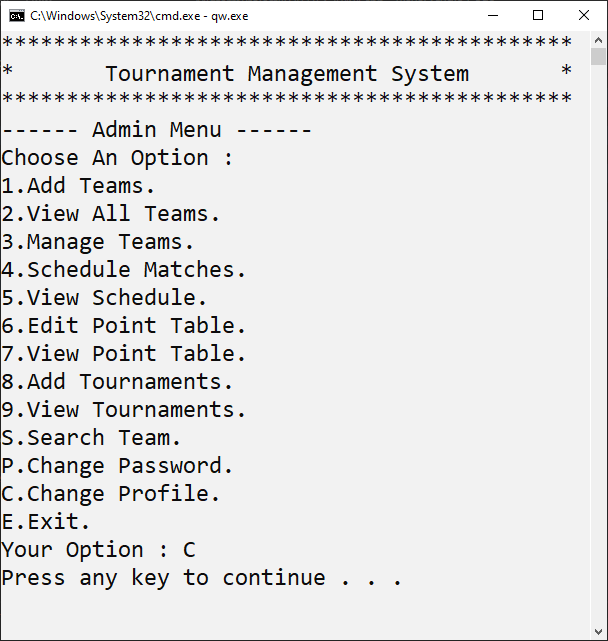
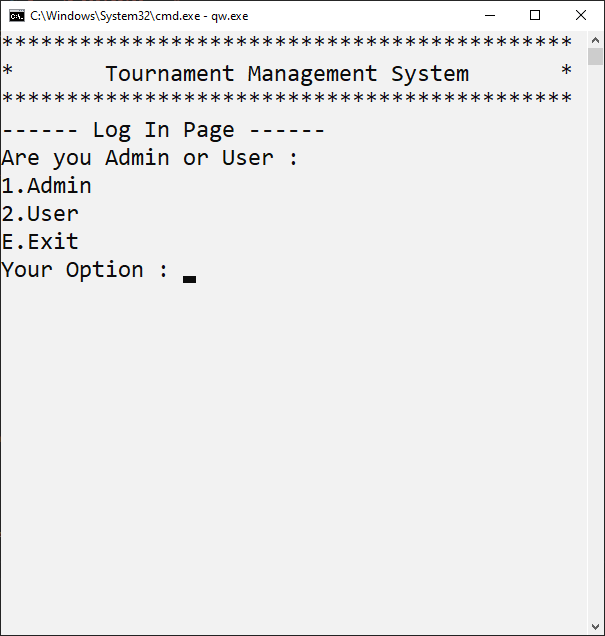
****

**1.9-View Tournaments:**

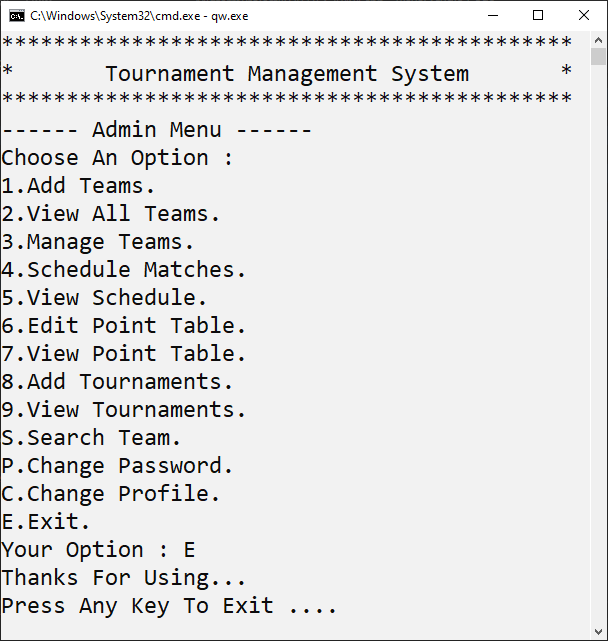
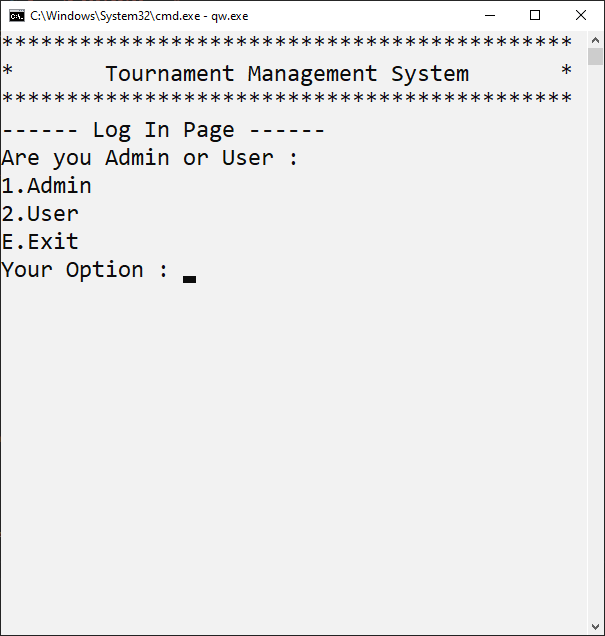
**1.10-Search Team:**

****

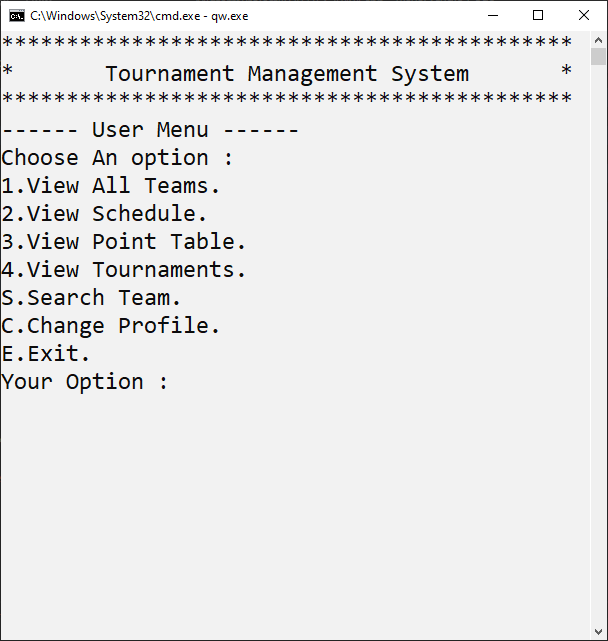
**1.11-Change Password:**

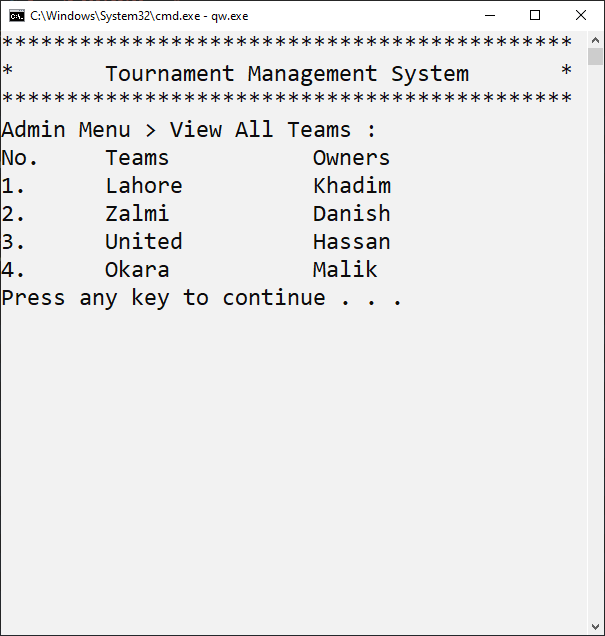
**1.12-Change Profile:**

**1.13-Exit:**

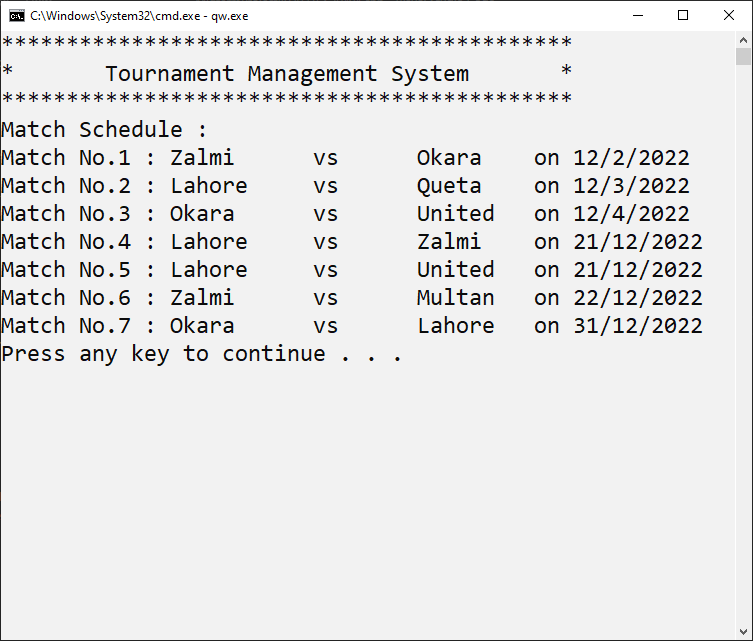
****

**2.User Menu:**

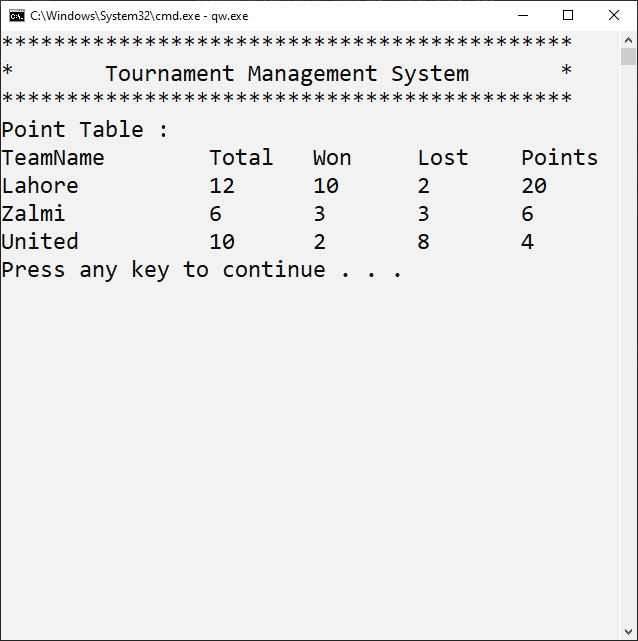
****

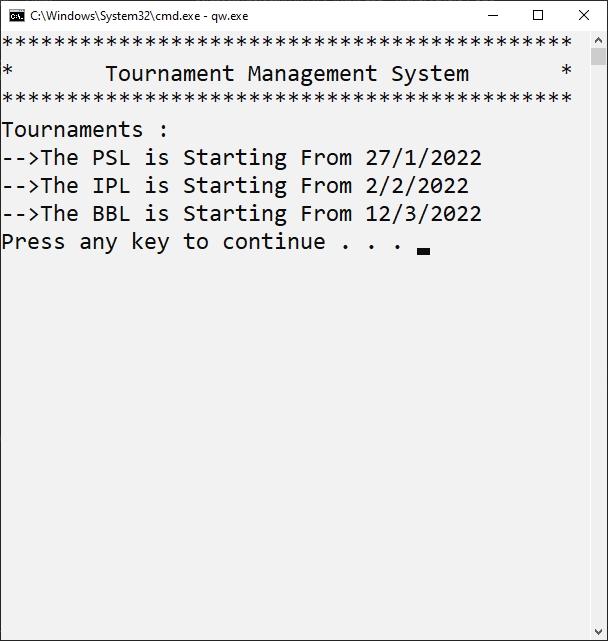
**2.1-View All Teams:**

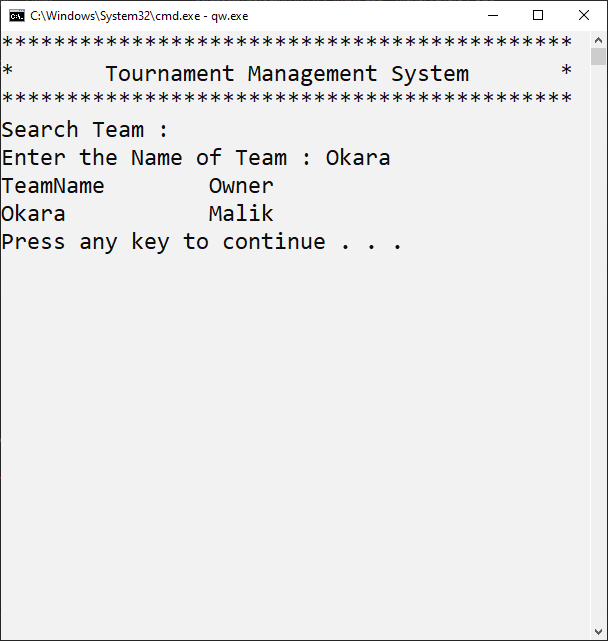
**2.2-View Schedule:**

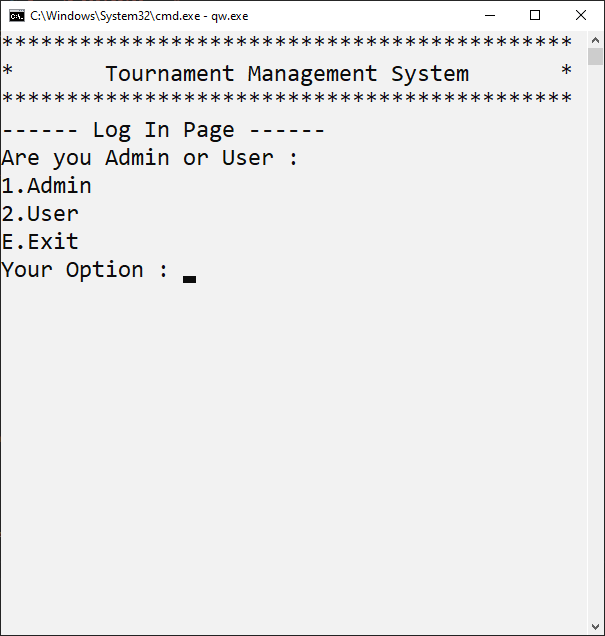
****

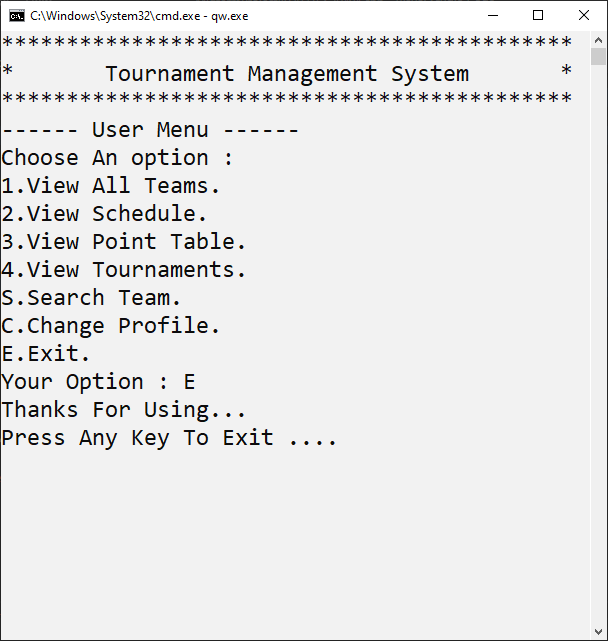
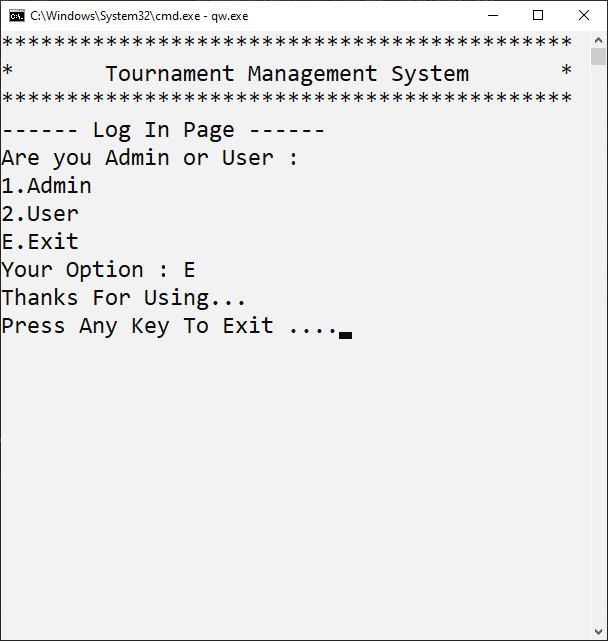
**2.3-View Point Table:**

****

**2.4-View Tournament:**

**2.5-Search Team:**

**2.6-Exit:**

**3.Exit:**

# 

**The End**