FEU Institute of Technology

COLLEGE OF ENGINEERING • COLLEGE OF COMPUTER STUDIES

Game Leaderboard

Algorithm – CS0007

Dr. Hadji Tejuco

Members:
Junio Adrian - Lead programmer
Opulencia, Kaiser - Error handling, Documentation

Submission date: 1/30/25

TABLE OF CONTENTS

Introduction

Purpose

Videogames has been a long standing entertainment, from the creation of “*Tennis for two”* way back in 1958 to the modern indie developed games and company backed studio of triple A games. The purpose of this program is to recreate those types of score leaderboard tracking and to learn how to optimize the system.

Objectives

- Make a game leaderboard that tracks the player and their score
- Use Priority Queue to manage the top 10 scores
- Use Linked list to store all player scores
- Use Array to store the top 10 scores for quick display

Scope

Included:

- Add/Update Player Scores
- Display top 10 player scores
- File saving for the top 10 player scores

Excluded:

- Load savefile to continue the leaderboard
- Simple game to go with the leaderboard
- GUI interface

Project Overview

Problem Statement

Technology evolved and videogames followed after but the core essence of games is to track scores between multiple players with the old Pac-Man in arcade machines or a single player game Sifu that tracks your score as you improve on clearing the different stages. How can the game leaderboard be improved upon?

Key Features

The program accepts new player names and scores and tracks them together to be listed onto a top 10 leaderboard that is in descending order to show the top highest score at the top and the lowest at the bottom.

Requirements Analysis

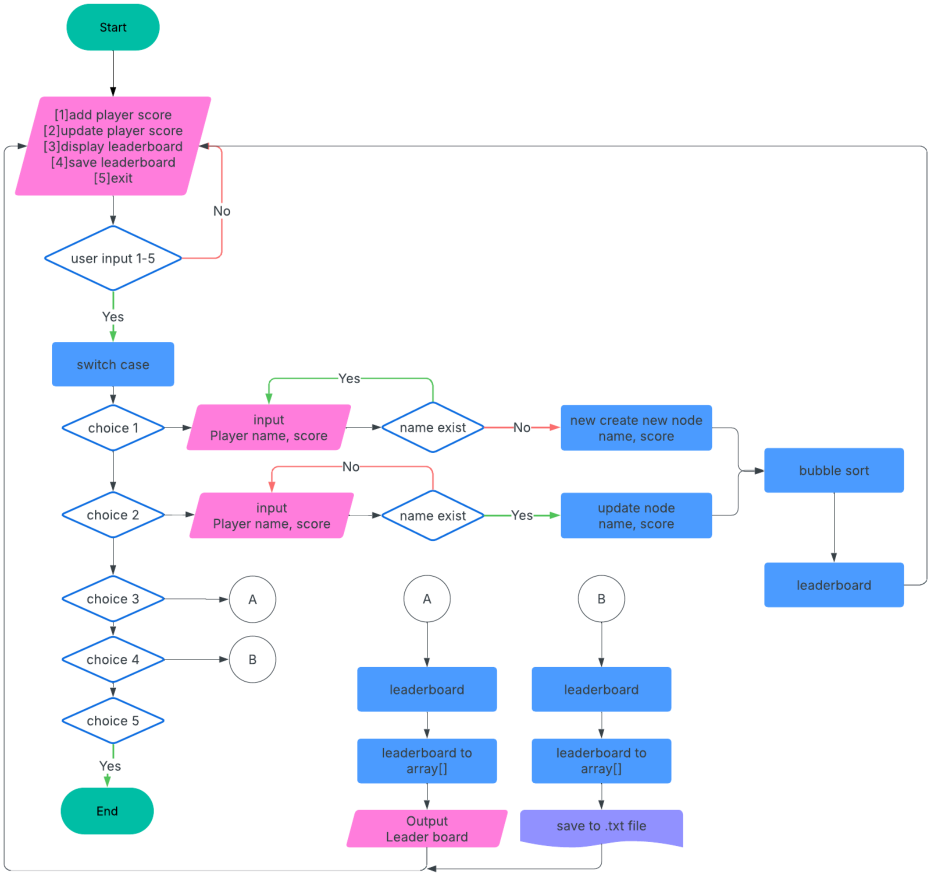
Functional Requirements

- When adding or updating the player score the input of the score should only be numbers and not other characters
- Must be in c++
- Use UDF, no special algorithms

Non-Functional Requirements

- Quick and responsive program that takes an average of 1.14seconds to compile
- User interface is clean and readable
- Only usable if user has c++ compiler to run the program
- Doesn’t have encryption to protect the data in the saved file

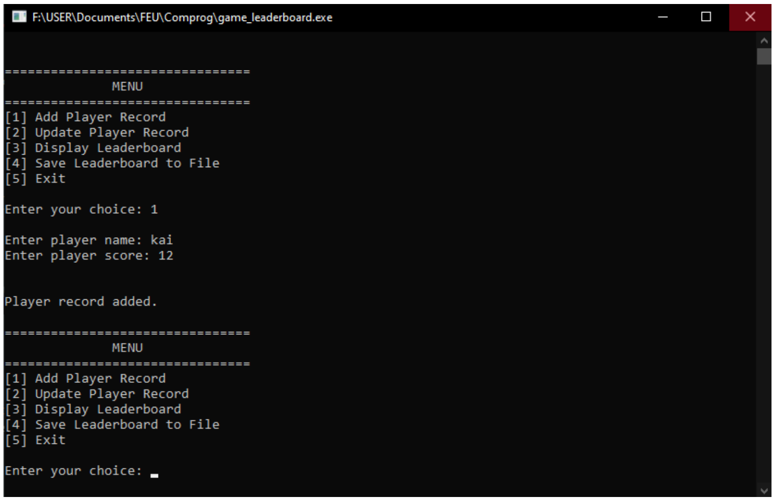
System Design



Implementation

Technologies Used

- Language uses is C++
- Iostream for input output, fstream for file handling, and string for string manipulation
- DevC++, onlineGDB, Github



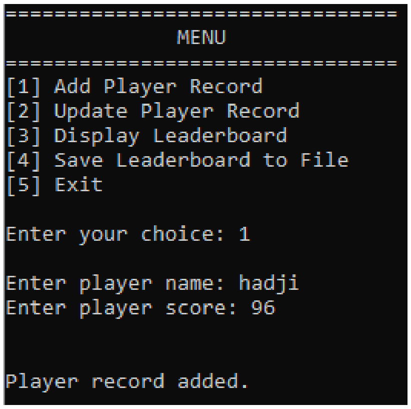
F:\USER\Documents\FEU\Comprog\game_leaderboard.exe

```
================================
              MENU
================================
[1] Add Player Record
[2] Update Player Record
[3] Display Leaderboard
[4] Save Leaderboard to File
[5] Exit

Enter your choice: 1

Enter player name: kai
Enter player score: 12


Player record added.

================================
              MENU
================================
[1] Add Player Record
[2] Update Player Record
[3] Display Leaderboard
[4] Save Leaderboard to File
[5] Exit

Enter your choice: _
```

Testing

case 1: Adding player name & score

Enter choice: 1 add player record
Enter player name: hadji
Enter player score: 96

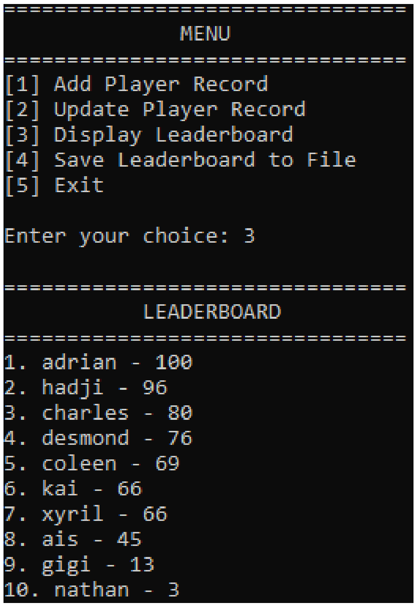


```
================================
              MENU
================================
[1] Add Player Record
[2] Update Player Record
[3] Display Leaderboard
[4] Save Leaderboard to File
[5] Exit

Enter your choice: 1

Enter player name: hadji
Enter player score: 96


Player record added.
```

The user when selecting the first option should be able to input the name of the player and the score

Case 2:display leaderboard

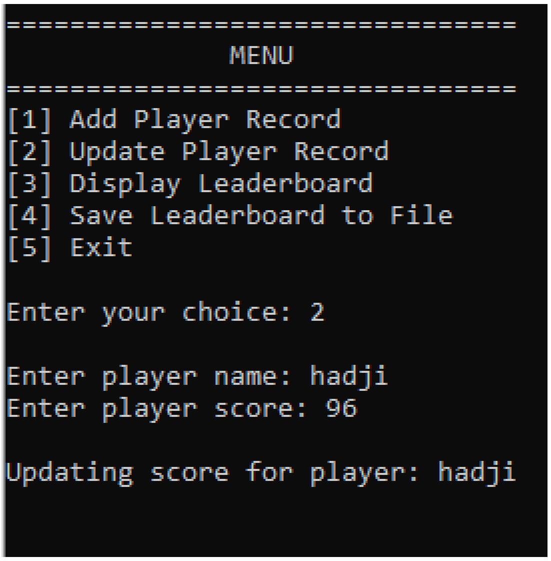
Enter choice :3



```
================================
              MENU
================================
[1] Add Player Record
[2] Update Player Record
[3] Display Leaderboard
[4] Save Leaderboard to File
[5] Exit

Enter your choice: 3

================================
           LEADERBOARD
================================
1. adrian - 100
2. hadji - 96
3. charles - 80
4. desmond - 76
5. coleen - 69
6. kai - 66
7. xyril - 66
8. ais - 45
9. gigi - 13
10. nathan - 3
```

Simple output and should only be the list of the players and their score in the leaderboard in a descending

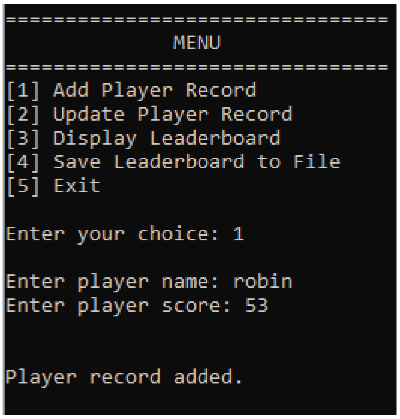
Case 3: Updating player score



```
================================
              MENU
================================
[1] Add Player Record
[2] Update Player Record
[3] Display Leaderboard
[4] Save Leaderboard to File
[5] Exit

Enter your choice: 2

Enter player name: hadji
Enter player score: 96

Updating score for player: hadji
```

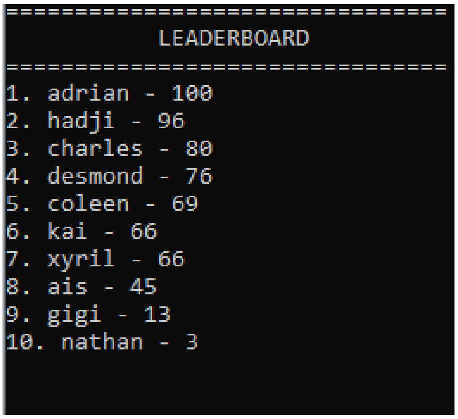
It asks the user to enter the player name and look for it in the leaderboard, if the player does not exist then prompting that it doesn't exist. If it exists then it will update the score to the new one.

Case 4: Add 11th player to replace a player in the leaderboard



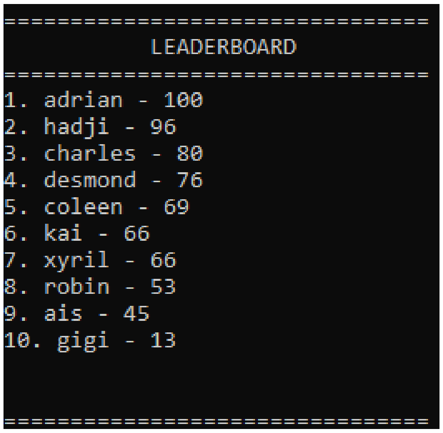
```
================================
              MENU
================================
[1] Add Player Record
[2] Update Player Record
[3] Display Leaderboard
[4] Save Leaderboard to File
[5] Exit

Enter your choice: 1

Enter player name: robin
Enter player score: 53


Player record added.
```

Before:



```
=================================
           LEADERBOARD
=================================
1. adrian - 100
2. hadji - 96
3. charles - 80
4. desmond - 76
5. coleen - 69
6. kai - 66
7. xyril - 66
8. ais - 45
9. gigi - 13
10. nathan - 3
```

After:

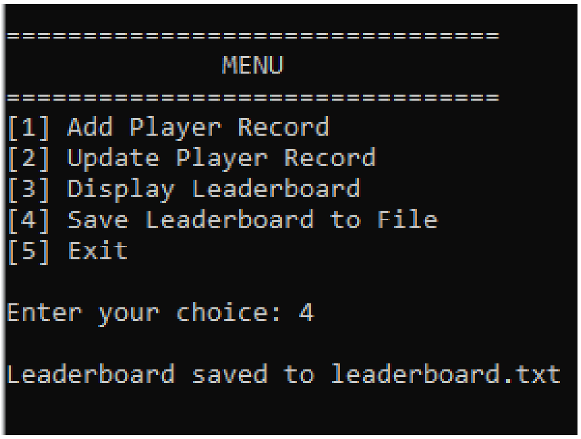


```
=================================
           LEADERBOARD
=================================
1. adrian - 100
2. hadji - 96
3. charles - 80
4. desmond - 76
5. coleen - 69
6. kai - 66
7. xyril - 66
8. robin - 53
9. ais - 45
10. gigi - 13


=================================
```

When a new player is added it the program will determine if the new player is in the top 10, if it is then the lowest player in the top 10 will be removed from the array.

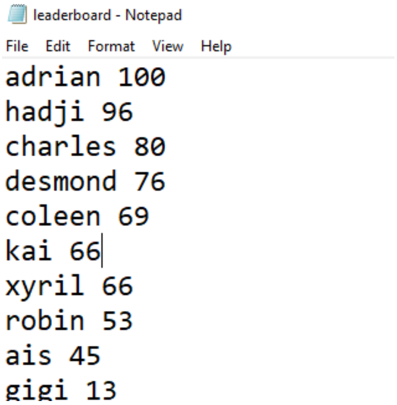
Case 5: Save into txt file

Enter choice: 4



```
================================
              MENU
================================
[1] Add Player Record
[2] Update Player Record
[3] Display Leaderboard
[4] Save Leaderboard to File
[5] Exit

Enter your choice: 4

Leaderboard saved to leaderboard.txt
```

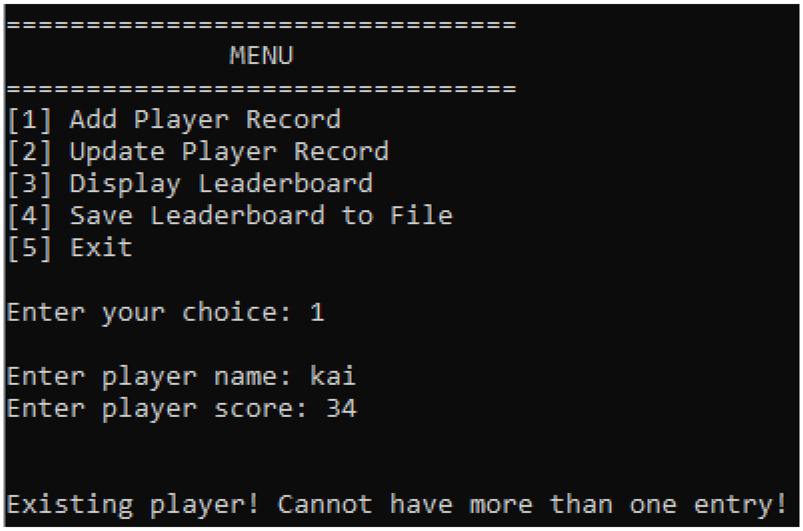
Text file should be in the same folder of the executable



```
adrian 100
hadji 96
charles 80
desmond 76
coleen 69
kai 66
xyril 66
robin 53
ais 45
gigi 13
```

With file handling the stored array of top 10 players will be printed on to each line on to a text file

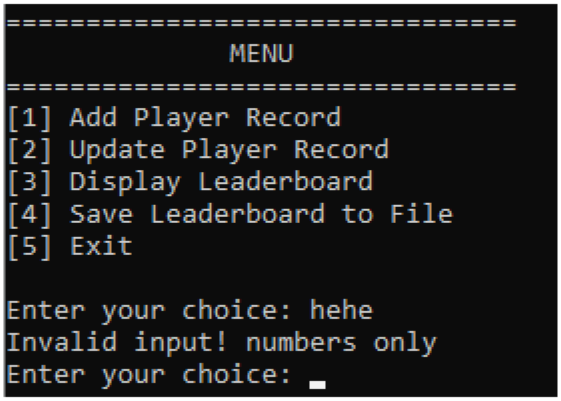
Case 6: Adding player that already have a record



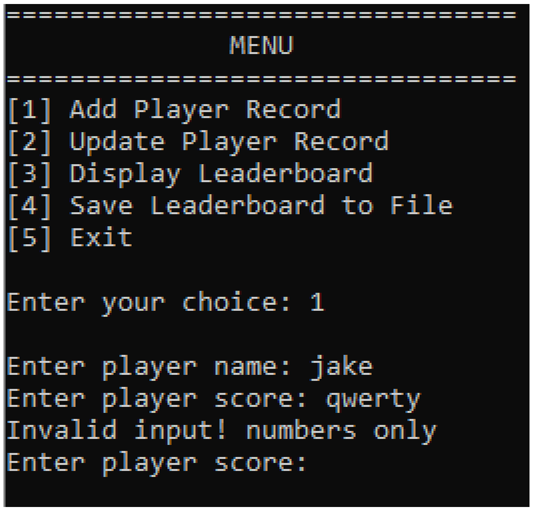
```
================================
              MENU
================================
[1] Add Player Record
[2] Update Player Record
[3] Display Leaderboard
[4] Save Leaderboard to File
[5] Exit

Enter your choice: 1

Enter player name: kai
Enter player score: 34


Existing player! Cannot have more than one entry!
```

When adding a new player in the record if the new name is identical to the stored name then the program will not allow that player name to be used prompting a different name.

Case 7: Error handling (user input mismatch)



```
================================
             MENU
================================
[1] Add Player Record
[2] Update Player Record
[3] Display Leaderboard
[4] Save Leaderboard to File
[5] Exit

Enter your choice: hehe
Invalid input! numbers only
Enter your choice: _
```



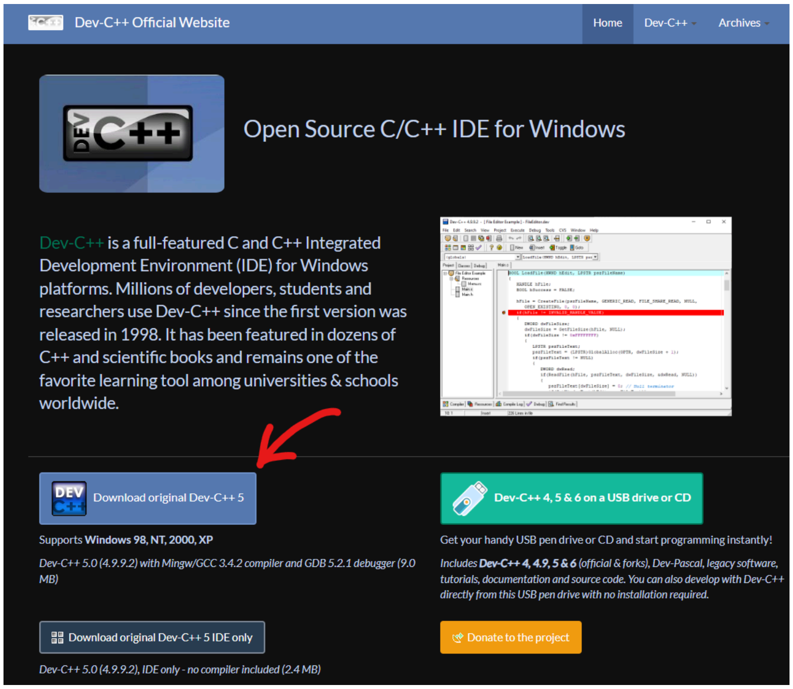
```
================================
             MENU
================================
[1] Add Player Record
[2] Update Player Record
[3] Display Leaderboard
[4] Save Leaderboard to File
[5] Exit

Enter your choice: 1

Enter player name: jake
Enter player score: qwerty
Invalid input! numbers only
Enter player score:
```

Most of the error handling is mismatch of user input from the required input

User Manual

Installation Guide

1. Install Dev C++ and follow their installation (https://www.bloodshed.net)



Open Source C/C++ IDE for Windows

Dev-C++ is a full-featured C and C++ Integrated Development Environment (IDE) for Windows platforms. Millions of developers, students and researchers use Dev-C++ since the first version was released in 1998. It has been featured in dozens of C++ and scientific books and remains one of the favorite learning tool among universities & schools worldwide.

Download original Dev-C++ 5
Dev-C++ 4, 5 & 6 on a USB drive or CD

Supports Windows 98, NT, 2000, XP

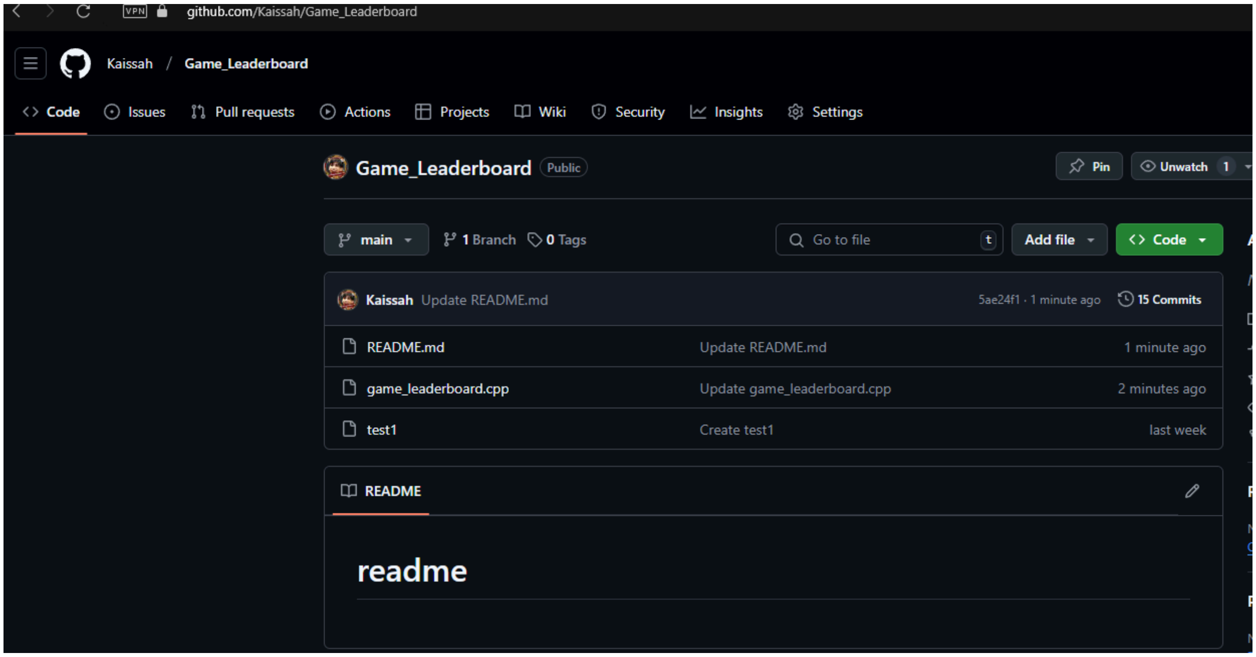
Dev-C++ 5.0 (4.9.9.2) with Mingw/GCC 3.4.2 compiler and GDB 5.2.1 debugger (9.0 MB)

Get your handy USB pen drive or CD and start programming instantly!

Includes Dev-C++ 4, 4.9, 5 & 6 (official & forks), Dev-Pascal, legacy software, tutorials, documentation and source code. You can also develop with Dev-C++ directly from this USB pen drive with no installation required.

Dev-C++ 5.0 (4.9.9.2), IDE only - no compiler included (2.4 MB)

2. Go to https://github.com/Kaissah/Game_Leaderboard



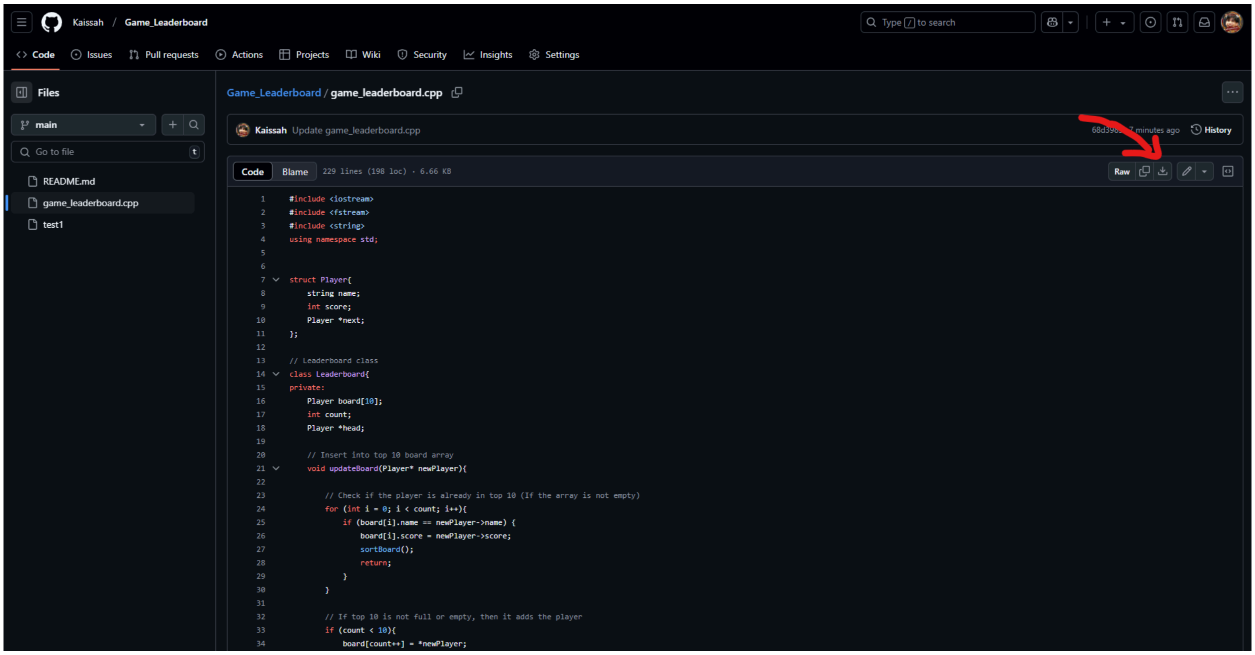
Kaissah Update README.md 5ae24f1 · 1 minute ago 15 Commits

README.md	Update README.md	1 minute ago
game_leaderboard.cpp	Update game_leaderboard.cpp	2 minutes ago
test1	Create test1	last week

README

readme

3. Select “game_leaderboard.cpp” and download the files

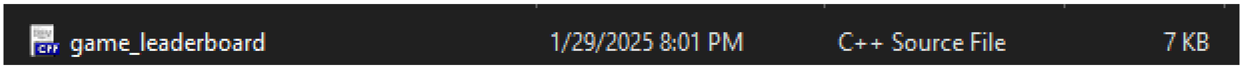


Kaissah Update game_leaderboard.cpp 68d398… 7 minutes ago

229 lines (198 loc) · 6.66 KB

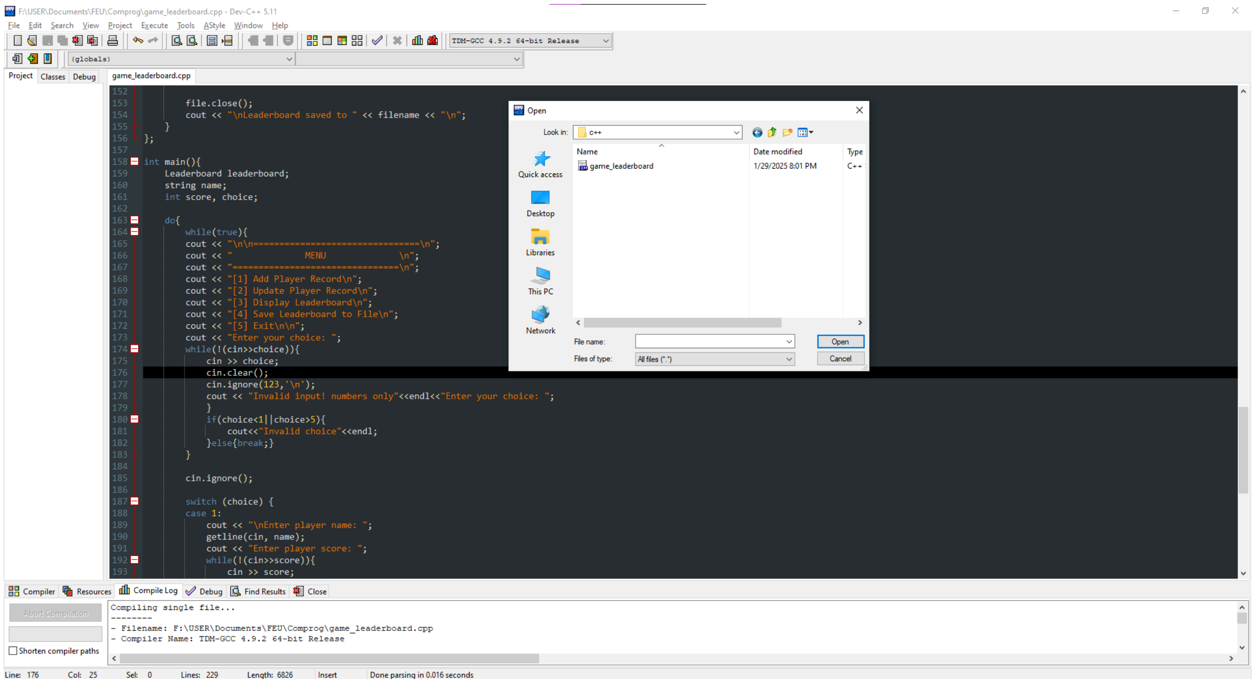
```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;


struct Player{
    string name;
    int score;
    Player *next;
};

// Leaderboard class
class Leaderboard{
private:
    Player board[10];
    int count;
    Player *head;

    // Insert into top 10 board array
    void updateBoard(Player* newPlayer){

        // Check if the player is already in top 10 (If the array is not empty)
        for (int i = 0; i < count; i++){
            if (board[i].name == newPlayer->name) {
                board[i].score = newPlayer->score;
                sortBoard();
                return;
            }
        }

        // If top 10 is not full or empty, then it adds the player
        if (count < 10){
            board[count++] = *newPlayer;
```

4. Select file location for the download



game_leaderboard	1/29/2025 8:01 PM	C++ Source File	7 KB

5. Open devC++ > open files > game_leaderboard.cpp



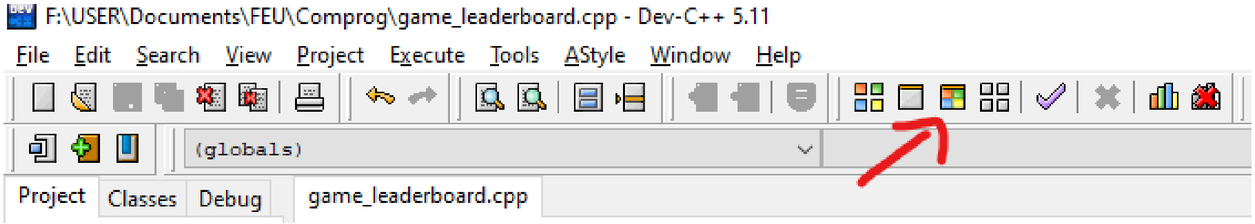
F:\USER\Documents\FEU\Comprog\game_leaderboard.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug game_leaderboard.cpp
Open
Look in: c++
Quick access
Desktop
Libraries
This PC
Network
Name
Date modified
Type
game_leaderboard
1/29/2025 8:01 PM
C++
File name:
Files of type: All files (*.*)
Open
Cancel
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
□ Shorten compiler paths

```
        file.close();
        cout << "\nLeaderboard saved to " << filename << "\n";
    }
};

int main(){
    Leaderboard leaderboard;
    string name;
    int score, choice;

    do{
        while(true){
        cout << "\n\n================================\n";
        cout << "                MENU              \n";
        cout << "================================\n";
        cout << "[1] Add Player Record\n";
        cout << "[2] Update Player Record\n";
        cout << "[3] Display Leaderboard\n";
        cout << "[4] Save Leaderboard to File\n";
        cout << "[5] Exit\n\n";
        cout << "Enter your choice: ";
        while(!(cin>>choice)){
            cin >> choice;
            cin.clear();
            cin.ignore(123,'\n');
            cout << "Invalid input! numbers only"<<endl<<"Enter your choice: ";
            }
            if(choice<1||choice>5){
                cout<<"Invalid choice"<<endl;
            }else{break;}
        }

        cin.ignore();

        switch (choice) {
        case 1:
            cout << "\nEnter player name: ";
            getline(cin, name);
            cout << "Enter player score: ";
            while(!(cin>>score)){
                cin >> score;
```

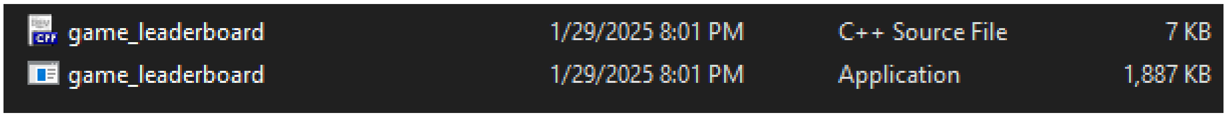
```
Compiling single file...
--------
- Filename: F:\USER\Documents\FEU\Comprog\game_leaderboard.cpp
- Compiler Name: TDM-GCC 4.9.2 64-bit Release
```

Line: 176 Col: 25 Sel: 0 Lines: 229 Length: 6826 Insert Done parsing in 0.016 seconds

6. Run and compile the program by pressing F11 or the button. You should have a .exe file in the same file location



F:\USER\Documents\FEU\Comprog\game_leaderboard.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug game_leaderboard.cpp



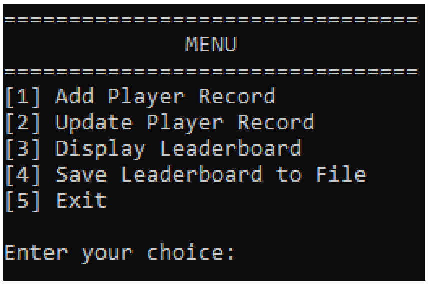
game_leaderboard	1/29/2025 8:01 PM	C++ Source File	7 KB
game_leaderboard	1/29/2025 8:01 PM	Application	1,887 KB

7. Application program is ready to use!

Usage Instructions

Start up

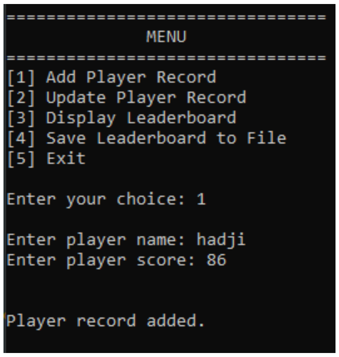
When you open the .exe executable file you should be greeted with this user interface that will show you the options to choose from.



```
================================
              MENU
================================
[1] Add Player Record
[2] Update Player Record
[3] Display Leaderboard
[4] Save Leaderboard to File
[5] Exit

Enter your choice:
```

[1] Add Player Record

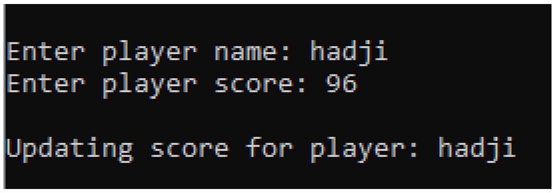
When adding a new player score the program will ask for player name and player score, the input on the player name is not restricted unlike the player score that only strictly collects numerical inputs. If the user input the wrong type it will be prompted to retry the attempt on adding the score



```
================================
              MENU
================================
[1] Add Player Record
[2] Update Player Record
[3] Display Leaderboard
[4] Save Leaderboard to File
[5] Exit

Enter your choice: 1

Enter player name: hadji
Enter player score: 86


Player record added.
```

[2] Update Player Record

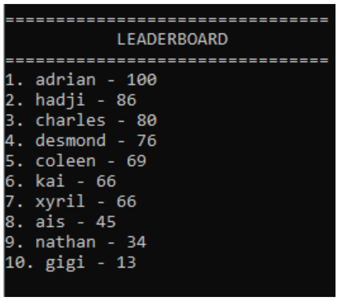
When updating the player score the player should initially exist in the leaderboard if the name of the player is not found then that mean that the player does not exist thus need to go to the **[1] Add player Record** menu to add the player. If the player exist then the old score will be overwritten with the new score updating the user that the player has been updated



```
Enter player name: hadji
Enter player score: 96

Updating score for player: hadji
```

[3] Display Leaderboard

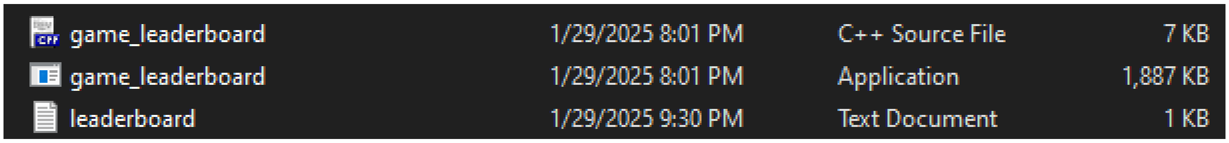
Upon selecting the 3rd option, display leaderboard the program will showcase the top 10 players in the leaderboard as shown the the figure below



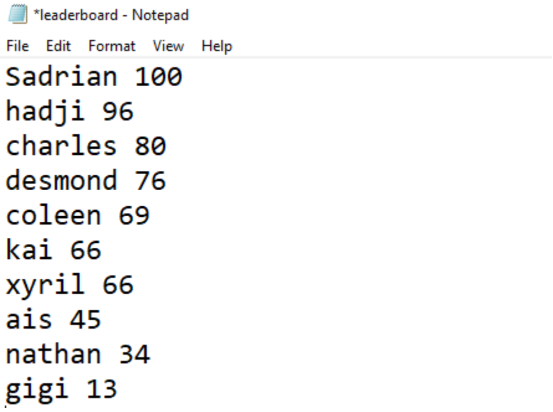
```
================================
          LEADERBOARD
================================
1. adrian - 100
2. hadji - 86
3. charles - 80
4. desmond - 76
5. coleen - 69
6. kai - 66
7. xyril - 66
8. ais - 45
9. nathan - 34
10. gigi - 13
```

[4] Save Leaderboard to File

When this option is selected the program will create a text file named leaderboard.txt that has the latest leaderboard scores and it will be saved where the file location of the executable.



```
game_leaderboard     1/29/2025 8:01 PM     C++ Source File        7 KB
game_leaderboard     1/29/2025 8:01 PM     Application        1,887 KB
leaderboard          1/29/2025 9:30 PM     Text Document          1 KB
```

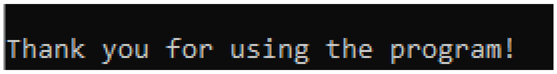


```
Sadrian 100
hadji 96
charles 80
desmond 76
coleen 69
kai 66
xyril 66
ais 45
nathan 34
gigi 13
```

Contents of the file created

[5] Exit

After you are done with the program you may select **[5] Exit** to close the program.



```
Thank you for using the program!
```

Challenges and Solutions

There were several problems we encountered. The earliest one is traversing through the list to organize it in a descending order and to play where the new player will be placed in the list. Next was in error handling where the program would infinite loop if the user input the wrong data type in the input. Lastly there were several times that the notion of creating a mini game in the program like a rock-paper-scissors that will keep track of the players score and will store it once the win streak ends.

How the solution is implemented is that with the sorting of the scoreboard, the priority queue that is used is the bubble sort to organize each player node to its corresponding score. Second problem's solution is to use a while loop that checks and repeats its prompt until the user input the desired input. Lastly it would take too long to create an additional minigame with the program due to time constraints but it would be wonderful to tackle the minigame creation as another project with the backbones of the leaderboard system of this project, it would be easily implemented onto the other project.

Future Enhancements

Future improvements could be in the program that there should be a feature that save files could be loaded back into the system after restarting the application. The program should also have a Graphical User Interface for the streamline experience for the user. Lastly there could be a faster sorting algorithm that can be implemented, for the mean time the player count is small but when it grows bigger the old sorting algorithm won't be able to handle that much load.

Conclusion

In the end the program runs as intended from organizing players with their scores to the creation of a save file. The program takes in player name and player score to be sorted into the top 10 players in the leaderboard with every addition of a new player an old player gets moved down or up the leaderboard. During the creation of the project the learning point is that the value of a sorting algorithm can streamline the process of the organizing of multiple players, and how linked lists should be used in the majority of the project for it dynamically changing size depending on the runtime of the user. Thinking outside of the box is a big help not just in solving problems but improving onto an idea, e.g. this leaderboard project will not show it full potential if there are no games or system attached to it, and it will make the leaderboard value not really needed.

References

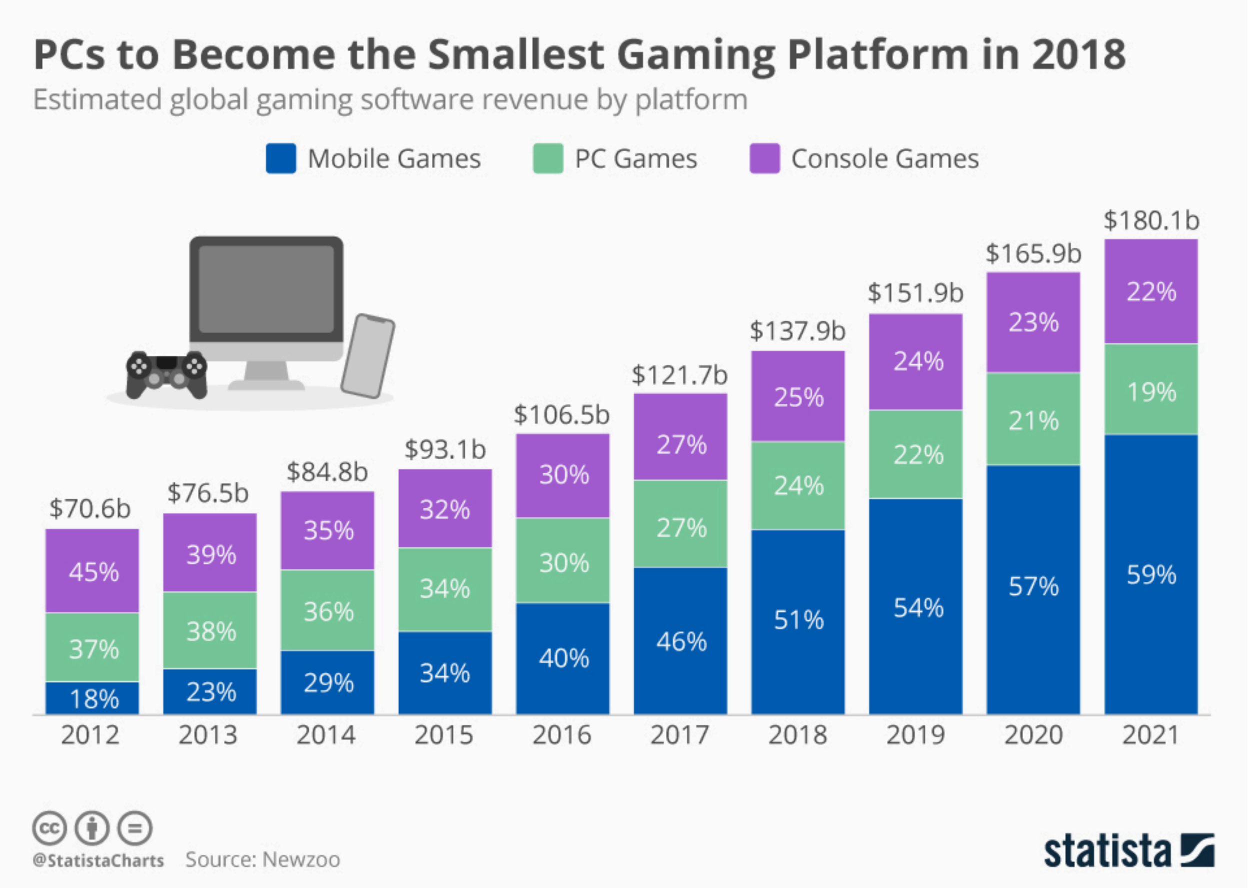
Tennis for Two (2025, January 24). In Wikipedia https://en.wikipedia.org/wiki/Tennis_for_Two

Research by Jörnmark, Axelsson, Ernkvist (2005) Wherever Hardware, There'll be Games: The Evolution of Hardware and Shifting Industrial Leadership in the Gaming Industry https://dl.digra.org/index.php/dl/article/download/205/205

Richter, F. (2018). PCs to Become the Smallest Gaming Platform in 2018 https://www.statista.com/chart/13789/worldwide-video-game-revenue-forecast/

Appendices

Appendix A: Additional diagrams or charts.



Source: Newzoo

Appendix B: Complete Code

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;


struct Player{
    string name;
    int score;
    Player *next;
```

```
};

// Leaderboard class
class Leaderboard{
private:
   Player board[10];
   int count;
   Player *head;

   // Insert into top 10 board array
   void updateBoard(Player* newPlayer){

      // Check if the player is already in top 10 (If the array is not empty)
      for (int i = 0; i < count; i++){
         if (board[i].name == newPlayer->name) {
            board[i].score = newPlayer->score;
            sortBoard();
            return;
         }
      }

      // If top 10 is not full or empty, then it adds the player
      if (count < 10){
         board[count++] = *newPlayer;
         sortBoard();
      } else{
         // Check if the new player's score is higher than the 10th player in the leaderboard
(triggers if and only if the array is full)
         if (newPlayer->score > board[count - 1].score) {
            board[count - 1] = *newPlayer;
            sortBoard();
         }
      }
   }

   // Sort top 10 array by score (descending) (Temporary: This should be the priority queue
part)
   void sortBoard() {
      for (int i = 0; i < count - 1; i++) {
         for (int j = 0; j < count - i - 1; j++) {
            if (board[j].score < board[j + 1].score ||
              (board[j].score == board[j + 1].score && board[j].name > board[j + 1].name)) {
               Player temp = board[j];
               board[j] = board[j + 1];
```

```
                    board[j + 1] = temp;
                }
            }
        }
    }

public:
    Leaderboard(){
        head = nullptr;
        count = 0;
    }

    // Add or update player
    void addPlayer(string name, int score){

        if (score < 0 || score > 100){
            cout << "Invalid score. Enter a score between 0-100.\n";  // Error handling for invalid
score.
            return;
        }

        Player* current = head;
        while (current != nullptr){
            if (current->name == name){
                cout << "\n\nExisting player! Cannot have more than one entry!";  // Error handling
for multiple entries
                return;
            }
            current = current->next;
        }

        // If list is empty or no matching player is found; add new player to the linked list.
        Player *newPlayer = new Player;
        newPlayer->name = name;
        newPlayer->score = score;
        newPlayer->next = head;
        head = newPlayer;

        updateBoard(newPlayer);
        cout << "\n\nPlayer record added.";
    }

    void updatePlayer(string name, int score){
        if (!head)
            cout << "\n\nNo records found. Player list is empty.";
        else{

            if (score < 0 || score > 100){
            cout << "Invalid score. Enter a score between 0-100.\n";  // Error handling for invalid
score.
            return;
            }

            // Search for the player in the linked list if the list is not empty.

            bool found = 0;
            Player *current = head;
            while (current != nullptr){
                if (current->name == name){
                    found = 1;
                    cout << "\nUpdating score for player: " << name << "\n";
                    current->score = score;
                    updateBoard(current);
                    return;
                }
                current = current->next;
            }

            if (found == 0)
            cout << "\n\nNo matching player found.";
        }
    }

    // Display top 10 players using the array.
    void displayBoard(){

        if (count == 0)
            cout << "\n\nLeaderboard is currently empty.";
        else{
            cout << "\n================================\n";
            cout << "          LEADERBOARD          \n";
            cout << "================================\n";
            for (int i = 0; i < count; i++) {
                cout << i + 1 << ". " << board[i].name << " - " << board[i].score << "\n";
            }
        }
    }

    // Save leaderboard to file
    void saveToFile(const char* filename){
        ofstream file(filename);
        if (!file) {
            cout << "Error: File does not exist.\n";     // Error handling for files
            return;
        }

       for (int i = 0; i < count; i++){
        file << board[i].name << " " << board[i].score << "\n";
        }


        file.close();
        cout << "\nLeaderboard saved to " << filename << "\n";
    }
};

int main(){
    Leaderboard leaderboard;
    string name;
    int score, choice;

    do{
                        while(true){
        cout << "\n\n=================================\n";
        cout << "              MENU              \n";
        cout << "=================================\n";
        cout << "[1] Add Player Record\n";
        cout << "[2] Update Player Record\n";
        cout << "[3] Display Leaderboard\n";
        cout << "[4] Save Leaderboard to File\n";
        cout << "[5] Exit\n\n";
        cout << "Enter your choice: ";
        while(!(cin>>choice)){
            cin >> choice;
            cin.clear();
            cin.ignore(123,'\n');
            cout << "Invalid input! numbers only"<<endl<<"Enter your choice: ";
            }
            if(choice<1||choice>5){
                        cout<<"Invalid choice"<<endl;
```

```
                }else{break;}
        }

cin.ignore();

switch (choice) {
case 1:
  cout << "\nEnter player name: ";
  getline(cin, name);
  cout << "Enter player score: ";
  while(!(cin>>score)){
    cin >> score;
    cin.clear();
    cin.ignore(123,'\n');
    cout << "Invalid input! numbers only"<<endl<<"Enter player score: ";
  }
  cin.ignore();
  leaderboard.addPlayer(name, score);
  break;
case 2:
  cout << "\nEnter player name: ";
  getline(cin, name);
  cout << "Enter player score: ";
  while(!(cin>>score)){
    cin >> score;
    cin.clear();
    cin.ignore(123,'\n');
    cout << "Invalid input! numbers only"<<endl<<"Enter player score: ";
  }
  cin.ignore();
  leaderboard.updatePlayer(name, score);
  break;
case 3:
  leaderboard.displayBoard();
  break;
case 4:
  leaderboard.saveToFile("leaderboard.txt");
  break;
case 5:
  cout << "\n\nThank you for using the program!";
  break;
default:
  cout << "Invalid choice. Please try again.\n";
}
```

```
    } while (choice != 5);

    return 0;
}
```