Schneider C. Jean-Pierre

Professor Francisco Mitropoulos

Capstone Project For Computer Science

Requirements Document

The Score Keeper

Brief Overview:

In this program, it would take in information in regards to multiple games played between a group of people. It would be the ideal way of keeping track of data from various game nights and/or gaming events. This program would be able to take in information from the user about the various rounds/matches of games at any time and keep a record of this. This would be great for tournaments, setting goals for prizes, or just to know who the best is at what.

Functional Requirements

(*Picture diagrams provided below according to the number in the list*)

1. Use Case: Open application (*requirements for when the user first opens up the application*). (See figure 1)

    1.1. Start menu is displayed.

    1.1.1. Create a new game.

    1.1.1.1. User should be presented with a button option to create a new game, once it is selected it should take them to a new window. (See 2).

    1.1.2. Create a new player profile.

1.1.2.1. User should be presented with a button option to create a new player profile, once it is selected it should take them to a new window. (See 5).

1.1.3. Player list

1.1.3.1. User should be presented with a button option to review the list of players created in the program and their stats, once it is selected it should take them to a new window. (See 6).

1.1.4. Game history

1.1.4.1. User should be presented with a button option to review the list of games created in the program and their stats, once it is selected it should take them to a new window. (See 7).

1.1.5. About

1.1.5.1. User should be presented with a button option to learn more about the program, once it is selected it should pop up with a new window. (See 8).

2. Use Case: Create a new game (*requirements for when the user selects the create a new game button option*). (See figure 2)

2.1. Input detection/collection

2.1.1. Fields such as the game's name, the number of rounds, the option of points per round or winner per round, the amount of points per round should be presented for the user to input, and once submitted those inputs should be assigned to separate variables. Most of these fields should be required.

2.2. Adding/removing of players

2.2.1. The user should be able to select from a list of already registered player profiles to add the players to the game. If the user changes their mind they should be able to remove a selected player. The players should be able to be remembered and track points for each. A minimum of one player should be implemented.

2.2.2. There should also be a button option that allows the user to create a new player profile, that will pop up and allow the user to input the data for the new player (similar to 5). Once completed the new player should appear in the player list.

2.3. Begin game tracking

2.3.1. Once all the information is inputted and the players are selected, the user should be able to select the begin game tracking button option (only if all required fields are met) and the game tracking should begin in a new window screen. (See 3).

3. Use Case: Begin game tracking (*requirements for when the user selects the begin game tracking button option*). (See figure 3)

3.1. Round tracking

3.1.1. The round number should be displayed and incremented after each round is completed by the user.

3.2. Point(s) tracking

3.2.1. The number of points a user gains per round should be able to be inputted by the user in a user-friendly way. If the game option is a winner(s) per round, the user would simply have to choose which user(s) won per round.

3.2.2. The points will be tracked by the program. It should not only track how many points the player has but it should also track who has the most points to determine the winner(s).

3.3. Next round/end game

3.3.1. There should be a button option that allows the user to finalise the information for the round and move to the next round. If it is the last round, the button would allow for the user to end the game. (See 4)

4. Use Case: End Game (*requirements for when the user selects the end game button option*). (See figure 4)

4.1. Display winner(s)

4.1.1. The winner(s) for the game should be displayed.

4.2. Display game stats

4.2.1. The information of the points should be displayed on the screen, listing the amount of points each player received per round.

4.3. Store game stats and winner(s)

4.3.1. The information of points distributed throughout the game should be stored onto an appropriate .txt file for storage. It should also be stored on a .txt file for each player so they retain their points information.

4.4. Create a new game

4.4.1. The user should be presented with a create a new game button option that will basically send them back to the create a new game screen (see 2).

4.5. Return to menu

4.5.1. The user should be presented with a return to menu button option that will basically send them back to the main menu (see 1).

5. Use Case: Create a new player profile (*requirements for when the user selects the create new player profile button option*). (See figure 5)

   5.1. Input detection/collection

      5.1.1. Fields such as the user's unique player name should be presented for the user to input, and once submitted those inputs should be assigned to separate variables. Most of these fields should be required.

      5.1.2. Must check if there is not another user with the same name in the system. Provides an error message if there is a similar name in the program.

      5.1.3. The information should also be stored and saved as a player profile where points and game information can be stored once the user submits it.

   5.2. Save/close button & create another player option

      5.2.1. Once the user creates the player profile, they should have the option to save the profile, then closing the window proceeding back to the main menu, or the option to immediately create a new player profile.

6. Use Case: View player list (*requirements for when the user selects the player list button option*). (See figure 6).

   6.1. Display players

      6.1.1. All player profiles created should be displayed via a list for the user to view.

   6.2. View player profile

      6.2.1. When a player profile is selected, their information should pop up on the screen along with their stats according to the points they've earned from all the games they've played.

6.3. Closing

6.3.1. Once the user is done viewing the players, there should be a close button option for them to select that will bring them back to the main menu screen.

7. Use Case: Game history (*requirements for when the user selects the game history button option*). (See figure 7)

7.1. Display previous games played on the program

7.1.1. All games played should be listed with dates played and its name.

7.2. Display more information

7.2.1. When a specific game is selected, their information should pop up on the screen along with their stats according to the points players earned each round.

7.3. Closing

7.3.1. Once the user is done viewing the games, there should be a close button option for them to select that will bring them back to the main menu screen.

8. Use Case: About (*requirements for when the user selects the about button option*). (See figure 8).

8.1. Display information about the program

8.1.1. Basically will provide the creators behind the program, the version, etc. displayed on the window.

8.2. Closing

8.2.1. Once the user is done viewing the information, there should be a close button option for them to select that will bring them back to the main menu screen.

9. Use Case: Closing the program (*requirements for when the user exits the program*).

   9.1. Closing

      9.1.1. When closing the program, it should retain all the information the user inputted and saved in .txt files for when they reopen the program next time.

      9.1.2. If the user decides to close the program before properly finishing any of the processes (player creation, game tracking, etc.) they should be prompted with an "are you sure, changes will not be saved" window.

Non-functional Requirements

- Must run on multiplatforms (Windows & Mac)

Pictures of Potential Design

Figure 1.

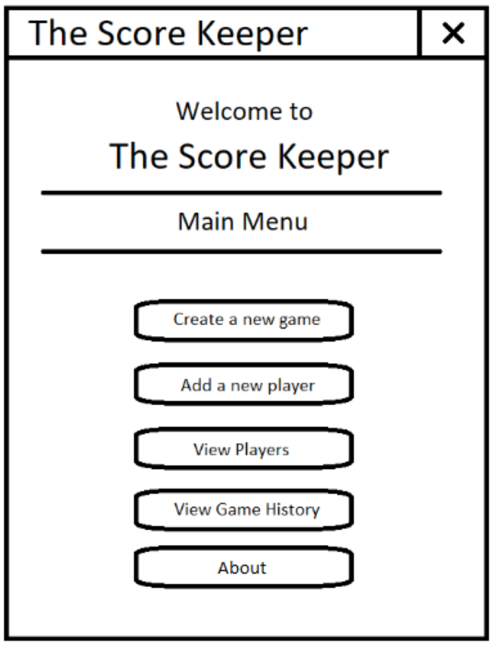
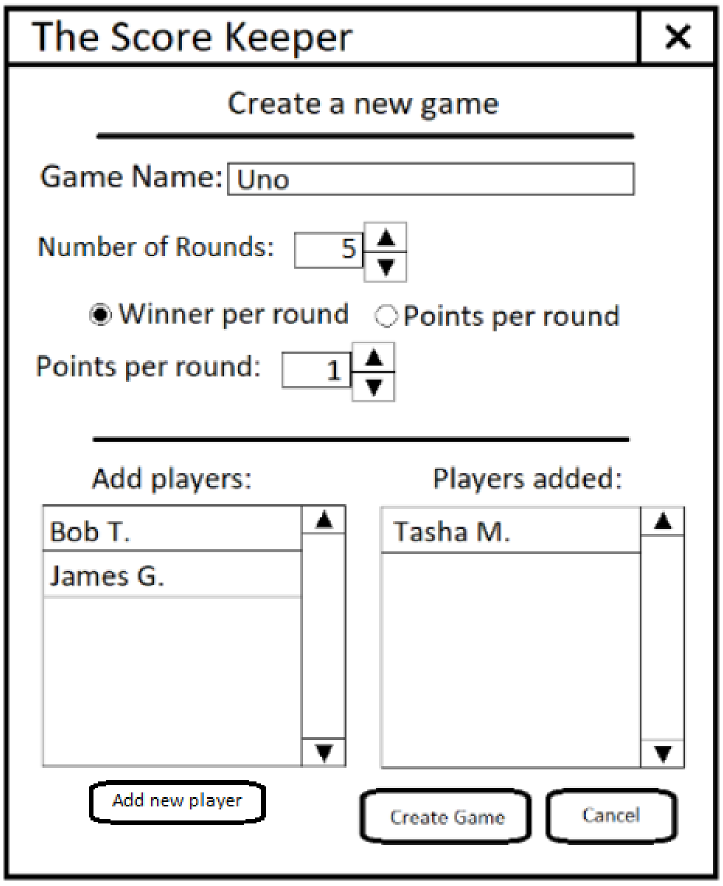


Figure 2.



Game Name: Uno

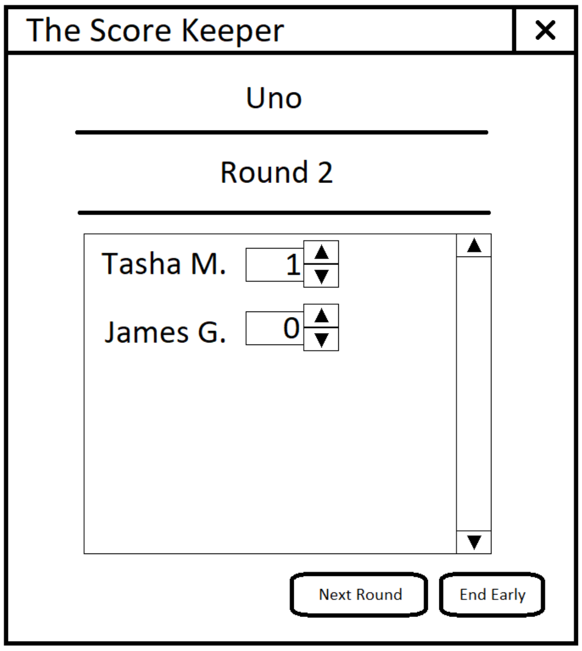
Number of Rounds: 5

[✓ Winner per round] Points per round

Points per round: 1

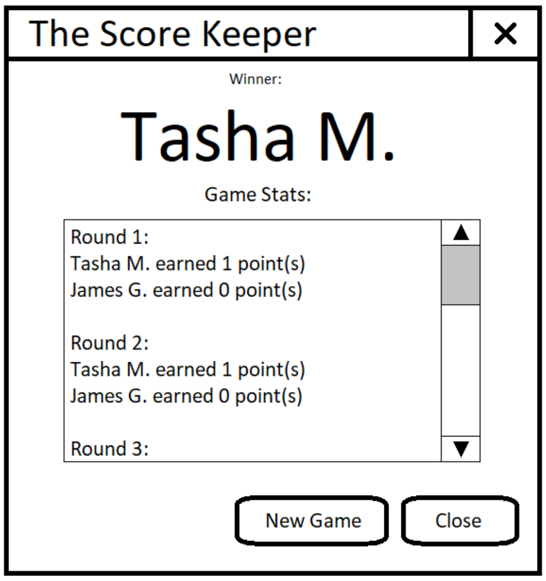
Add players:	Players added:
Bob T.	Tasha M.
James G.	

Figure 3.



Tasha M.	1
James G.	0

Figure 4.


The Score Keeper

Winner:

Tasha M.

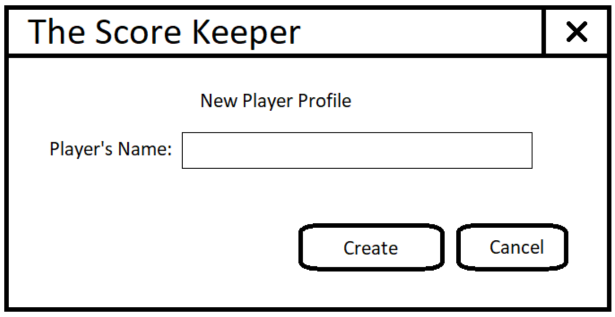
Game Stats:

Round 1:
Tasha M. earned 1 point(s)
James G. earned 0 point(s)

Round 2:
Tasha M. earned 1 point(s)
James G. earned 0 point(s)

Round 3:

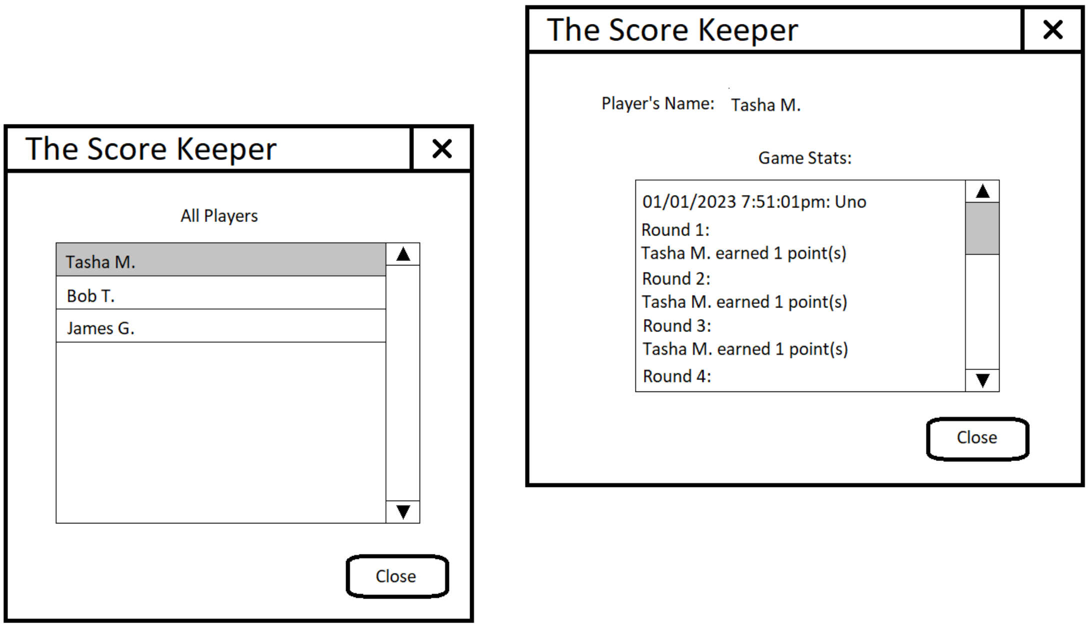
Figure 5.


The Score Keeper

New Player Profile

Player's Name:

Figure 6.



All Players

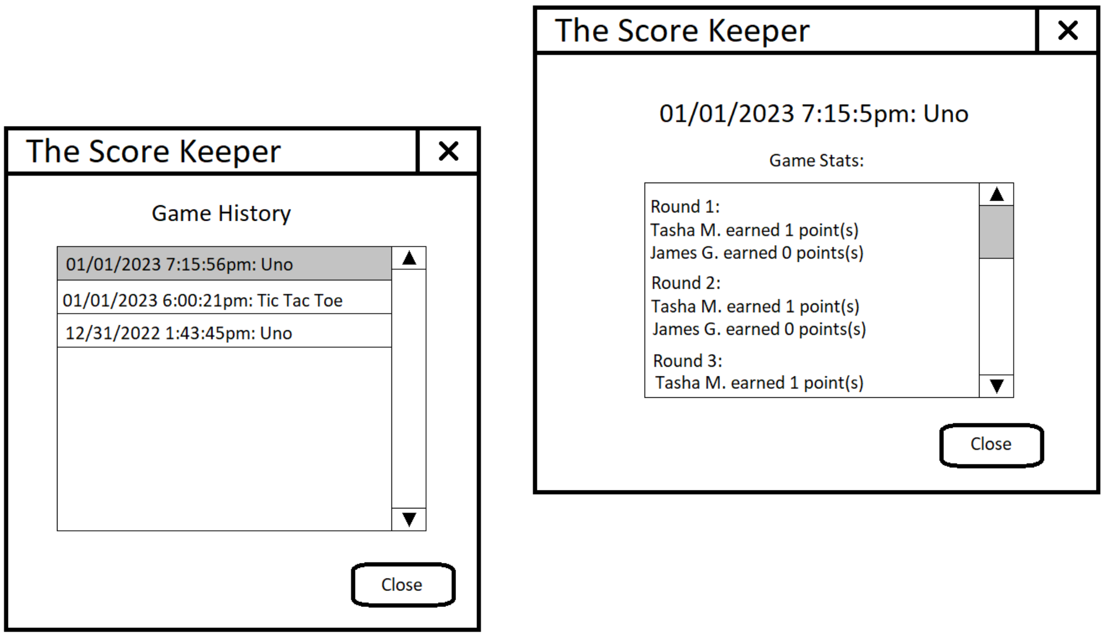
Tasha M.
Bob T.
James G.

Player's Name: Tasha M.

Game Stats:

```
01/01/2023 7:51:01pm: Uno
Round 1:
Tasha M. earned 1 point(s)
Round 2:
Tasha M. earned 1 point(s)
Round 3:
Tasha M. earned 1 point(s)
Round 4:
```

Figure 7.



Game History

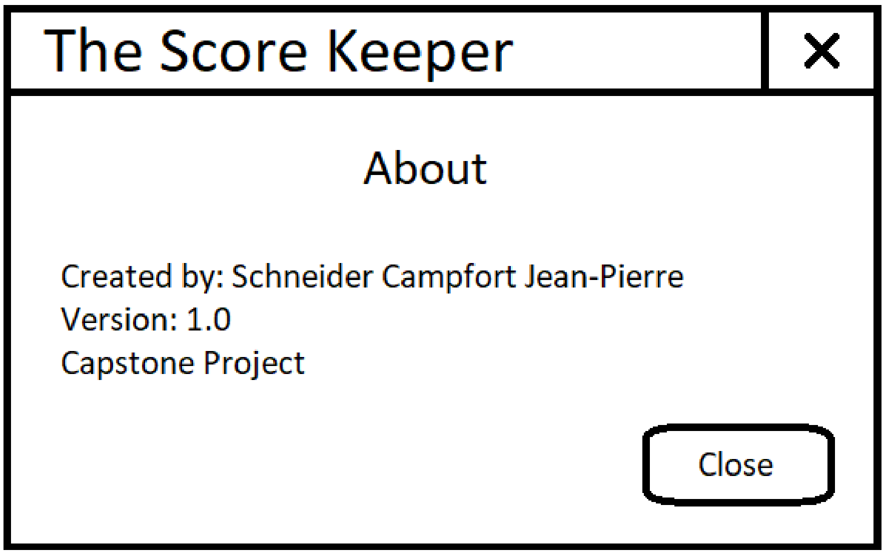
01/01/2023 7:15:56pm: Uno
01/01/2023 6:00:21pm: Tic Tac Toe
12/31/2022 1:43:45pm: Uno

01/01/2023 7:15:5pm: Uno

Game Stats:

```
Round 1:
Tasha M. earned 1 point(s)
James G. earned 0 points(s)
Round 2:
Tasha M. earned 1 point(s)
James G. earned 0 points(s)
Round 3:
Tasha M. earned 1 point(s)
```

Figure 8.


The Score Keeper

About

Created by: Schneider Campfort Jean-Pierre
Version: 1.0
Capstone Project