**Multiplayer Yahtzee**

**Final Report**

**Lil’ Yahtzee**

George Calvert, David Giacobbi, Henry Stone

Course: CPSC 224 - Software Development

# Introduction and Project Description

The following documentation is a summary of Lil’ Yahtzee’s efforts to plan, design, and implement a multiplayer Yahtzee program, utilizing Java and Java Swing. In addition to the source code itself, this report explains how our team approached and solved the requirements provided for multiplayer Yahtzee. Lastly, it will also include a discussion of what improvements and future developments can be made on the project.

The application that we are designing is a multiplayer Yahtzee. The goal of the game is to achieve the highest score out of all the other players. Yahtzee is a turn-based game where players roll a set of n-dice. Every turn, each player gets a chance to roll up to three times. After each roll, a player can choose to keep certain dice to attain the scoring lines on the scorecard. The game ends once every player has filled every scoring line in their scorecard.

The Yahtzee scorecard is divided into two sections: Upper Scoring and Lower Scoring. The upper scorecard is centered around collecting as many of the same side dice as possible. If the die has six sides, then there are six scoring opportunities on the player’s scorecard. The more of the same sided dice a player has, the larger number of points one can attain for that line. Like poker and other card-scored games, Yahtzee lower scoring lines involve patterns within the dice. The lower scorecard consists of a Three of a Kind, Four of a Kind, Full House, Small Straight, Large Straight, Yahtzee, and Chance lines. These lower scores are harder to attain, but the reward is greater with a larger number of points depending on the line. Every time a scoring line is filled in either the upper or lower scorecard, that line can no longer be used in future turns. The game ends once every player has used all their available scoring lines.

To implement these game objectives, a robust graphical user interface (GUI) is a necessity to build these ideas up in a visual manner. The GUI will be made up of four distinct screens: Welcome Screen, Make Players Screen, Yahtzee Screen, and End Game Screen. The Welcome Screen will configure the settings for the game (including number of players, number of dice, and number of sides). The Player Name Screen will let players set their names before starting the game. The Yahtzee Screen runs the bulk of the program, including each player’s dice rolls, scoring, and scorecard. Finally, once the scorecards are all filled, the End Game Screen will display the winner’s name, show the player rankings by grand total, and an option to play again.

# Team Members - Bios and Project Roles

David Giacobbi:

David is a computer science student interested in cybersecurity, app development, and staying active. His prior projects have included a data science analysis on his Apple Watch activity and a database application that helps users save and review travel packages for vacations. David’s skills include C++, Python, Java, SQL, and object-oriented programming. For this project his responsibilities include developing and designing the Yahtzee GUI, fixing bugs concerning the game’s internals, and creating visual presentation elements.

George Calvert:

George is a computer science student at Gonzaga University who is interested in software development, cyber security, and data science. Prior projects of his include developing machine learning models on his own Netflix viewing history to predict the number of shows watched given the type of weather and an application that allows users to develop and share flashcard sets. George’s skills include Python, C++, Java, SQL, and Jupyter Notebooks. His responsibilities for this project included developing the main algorithms and functions to compute and run the functional requirements of this application.

Henry Stone:  
  
 Henry is a computer science student interested in data science, software development, and music. Previous projects he has worked on include data science work on his music listening habits using API’s, and a texted based randomized adventure game. Henry’s skills include Python, C++, and Java. For this project his responsibilities included preliminary development, testing, and test development.

# Project Requirements

The following tables are Lil’ Yahtzee’s base project requirements. In addition to functional requirements, the team also attempted to create a few non-functional features that would further enhance the user’s game experience.

**Functional Requirements**:

|  |  |
| --- | --- |
| Part | Initial Display |
| Priority | High |
| Purpose | Welcome user, configure dice attributes |
| Inputs | Combo-Boxes to configure dice, Start button |
| Operators | User |
| Outputs | Dice attributes for the game |

|  |  |
| --- | --- |
| Part | Set Player Names |
| Priority | High |
| Purpose | Prompt users to enter names for their player |
| Inputs | Text Fields to hold users’ names, Game button |
| Operators | User |
| Outputs | User player names in String array |

|  |  |
| --- | --- |
| Part | Roll Dice |
| Priority | High |
| Purpose | Display rolled dice to screen in images, store dice values in an integer array list |
| Inputs | Roll Dice button |
| Operators | User |
| Outputs | Updated array list, dice images |

|  |  |
| --- | --- |
| Part | Display Current Scorecard |
| Priority | High |
| Purpose | Display score card of current player’s turn |
| Inputs | Display Current Score button |
| Operators | User |
| Outputs | None |

|  |  |
| --- | --- |
| Part | Choose Scoring Lines |
| Priority | High |
| Purpose | Select current user’s score for their turn, change their scorecard after selected |
| Inputs | Add Score button, Radio Buttons of each scoring line still available |
| Operators | User |
| Outputs | Current user’s updated score |

|  |  |
| --- | --- |
| Part | Switch Users (Multiplayer Functionality) |
| Priority | High |
| Purpose | Determine when the current player’s turn is over, start new turn for next player |
| Inputs | None (internal) |
| Operators | User |
| Outputs | None |

|  |  |
| --- | --- |
| Part | Final Leaderboard |
| Priority | High |
| Purpose | Show the final scores ranked on the End Game Screen |
| Inputs | None |
| Operators | User |
| Outputs | Ranked table of player scores |

# Non-Functional Requirements:

|  |  |
| --- | --- |
| Part | Text and Color Theme |
| Priority | Medium |
| Purpose | Create a visually pleasant color and font scheme |
| Inputs | None |
| Operators | User |
| Outputs | None |

|  |  |
| --- | --- |
| Part | Frame Layouts |
| Priority | High |
| Purpose | Display game elements in a robust and user-friendly manner |
| Inputs | None |
| Operators | User |
| Outputs | None |

|  |  |
| --- | --- |
| Part | Sound Effects |
| Priority | Low |
| Purpose | Create sound effects for button clicks to help transition frames better |
| Inputs | Game buttons |
| Operators | User |
| Outputs | Sound effects from .wav files |

# Solution Approach

**UML Class Diagram:**

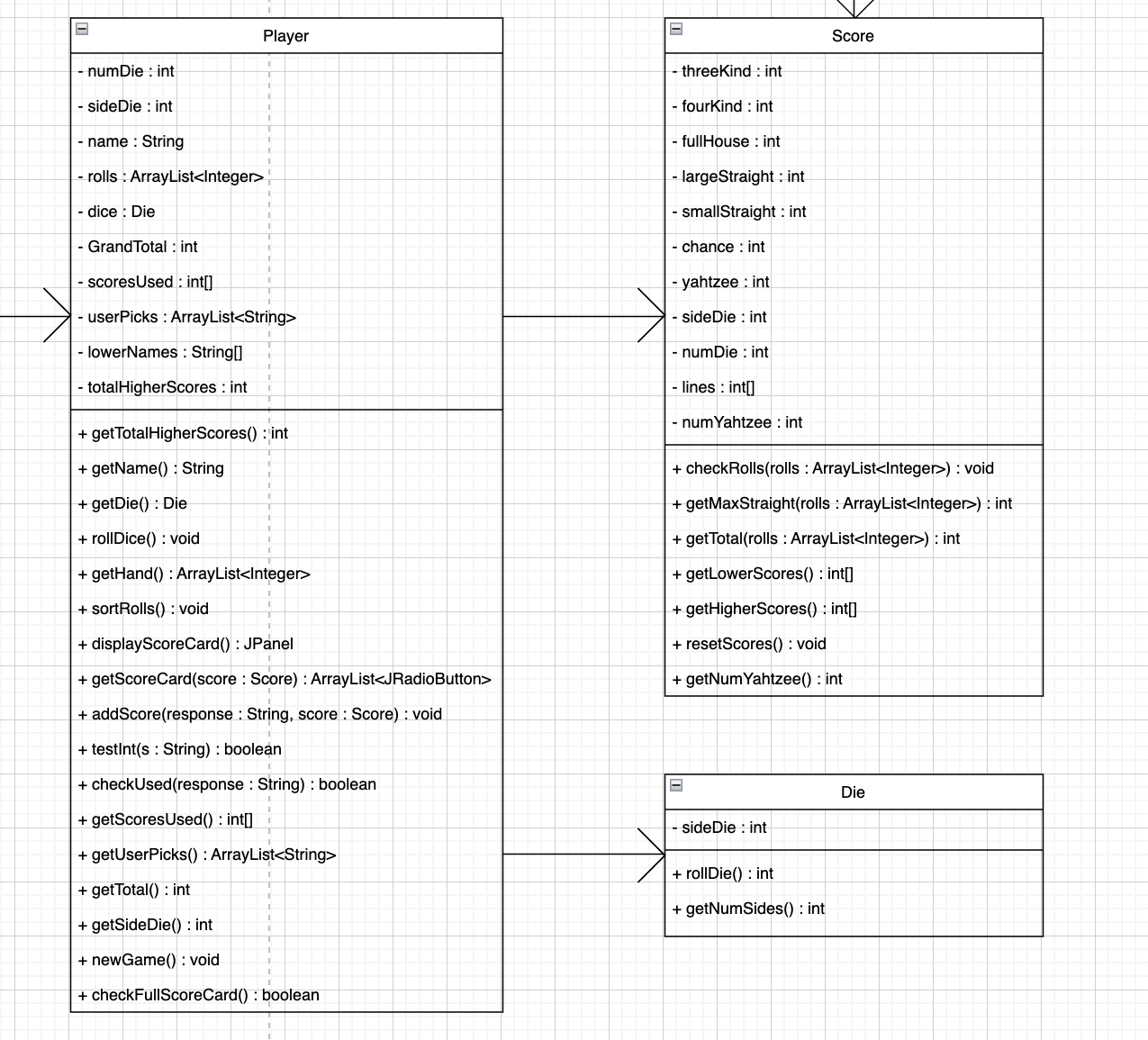
In order to maintain object-oriented principles, Lil’ Yahtzee approached multiplayer Yahtzee with a variation of the Model View Controller principle. This was done by separating the internal game functions from the Java Swing objects that display the game to the user.

Graphical user interface

Description automatically generated

Internal Game Objects:

The purpose of these objects is to run the actual Yahtzee game, under the hood. Their main functionality includes tracking player scores, rolling the dice, determining possible scoring lines after each turn, and generating a current scorecard.



Graphical User Interface Objects:

The purpose of these objects is to track the sequence of the game visually for the user. In addition to displaying any visual or including audio elements for Yahtzee, these objects manage the multiplayer functionality and act as the overseer for calling internal game objects.

Graphical user interface, application

Description automatically generated

**UML Sequence Diagrams:**

The two sequence diagrams below provide a brief overview of how the program traverses through objects during a full game of multiplayer Yahtzee. Though it cannot acknowledge loops, they will help clarify exactly how each object group interacts with each other during certain crucial moments in the game.

Internal Game Sequence:

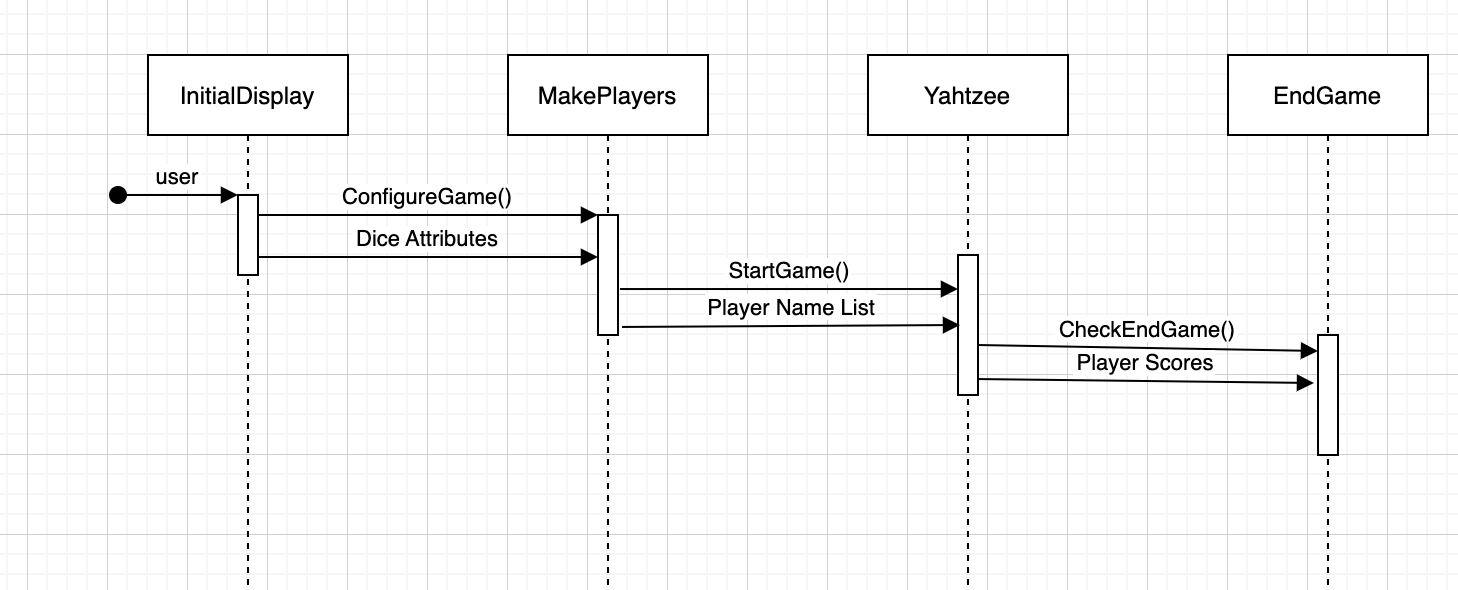
Since there are a lot of loops within the whole game. The following diagram how the game works under the hood for a singular turn.

Diagram

Description automatically generated

GUI Sequence:

The GUI sequence diagram is rather simple and informal. Its main goal is to convey how the GUI objects shift from one to another as well as what each object needs to continue on with the game.



# Test Plan

**Unit Testing**

Unit testing will be the main form of testing for all the game’s backend functions. As mentioned in the project overview, it is critical that the scoring and dice rolling are flawless in nature. Given that these two functional requirements determine the winner of the game, any kind of bug relating to these internals could effectively alter the outcome of the game.

To best utilize team resources, however, only critical functions relating to these two classes will be thoroughly checked for edge cases/special cases. Unit tests will only be completed on internal object functions. These tests can be found in the YahtzeeTest class in the team GitHub Repository.

**Integration Testing**

Integration testing will be important in ensuring that functions interact in the right manner with each other. An example of this type of test would be using the debugger in VS Code to check whether the array list that holds the current hand’s dice values are being accurately printed to the screen with the image icon function. Though code is not used to test these integrations, the team will utilize this test greatly during the development progress. Another important thing to note is that any merge requests will be reviewed by the group before adding it to the codebase to ensure there is always a functioning game in GitHub at all times.

Integration testing was utilized heavily during the GUI design process. Every time a new Swing element was added or functionality was added to one of the buttons, integration tests were quickly run to see if the button was functioning correctly before moving on to the next part of the game implementation.

**System Testing**

System testing will occur for the team after a primitive but functioning multiplayer Yahtzee has been built. Since these tests involve the game as a whole, it will be important to identify these bugs before creating the final release for the game. The main focus of this type of testing is to ensure that the GUI is running properly. The team wants to answer questions such as: What happens if a button is repeatedly pressed? What happens if a certain check box is never selected? What if a player does not enter a name?

Since the game can run despite these bugs, system testing is done last because it is intended to refine an already functioning game.

**Functional Testing**

Given the abundance of unit and integration tests that will be performed on critical functions, functional testing is considered to be a combination of these two tests.

**Performance Testing**

The team intends to create a game with a limited amount of opportunity for users to break the game with stress input. However, one test that will need to be completed is a check of how many characters a player can enter for their player’s name.

During performance testing, it was decided that default names would be provided for users that did not enter a username in their text box. Additionally, users that added a name that was more than 30 characters would only have the first 30 characters of their name.

Another performance test revealed that users could spam click the display scorecard button during their turn. To resolve this issue, George implemented a checking variable that would only allow for one scorecard to be displayed every turn.

**User Acceptance Testing**

User Acceptance will be a major part of system testing as it will give the team a good idea of what can be improved and fixed to best accommodate the users playing the game. To do this, a beta version of the game will be released to team member’s family, friends, and roommates. Team members will then return to the meeting with every player’s suggestion, bugs identified, and improvements.

During user acceptance testing, many minor bugs were quickly identified. Many of them were minor changes to how scoring worked so that it would work on any dice attribute combination. Additionally, an issue with checking the end of the game with multiple players was identified and resolved utilizing this type of test.

# Project Implementation Description

**Design Plan and Initial Approach:**

During the designing phase, Lil Yahtzee made sure to achieve functional requirements before shooting for enhancements to the game. The primary goal of the group was to get multiplayer functionally working, even if the GUI was very simple and not decorated.

Additionally, the group put heavy emphasis on ensuring that objects were well crafted and designed so that anyone could follow the program’s design with the aid of a couple of visual diagrams.

Ultimately, the initial approach was to set the functional goals to an attainable milestone that could be completed by the code deadline. Then, with any extra time and during the debug process, recommendations were to be implemented afterwards.

Below is the main Yahtzee frame plan that was drawn up as the group’s reference. The other project design screens can be found in the Appendix at the end of the document.

Diagram

Description automatically generated

A picture containing diagram

Description automatically generated

**Implementation:**

Implementation proved to be the most difficult part of the project for Lil Yahtzee. Despite the extensive plan details that were set in place, it was quickly revealed that more discussion about how to tackle certain functional requirements would require a complete class redesign. Moreover, it was difficult at first to exchange ideas because each programmer had their own unique idea of the approach to set up the objects. As a result, internal object implementation took longer than expected; furthermore, the group decided it would be best to put George in charge of the algorithm base to build the game, given his previous success with his own Yahtzee game.

While George worked on the algorithms under the hood, Henry took initiative in writing the team’s unit tests for the algorithms that were completed. Merge conflicts were often resolved quickly as the group ensured that messaging about what was being changed was conveyed clearly before starting their branch. Lastly, David developed a well-designed GUI that would be easy to read and follow.

Below are a couple images of the actual multiplayer Yahtzee final program. Though they have some differences compared to the initial design, the structure and flow of the game still follow the same sequence.

Graphical user interface, application

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

**System Testing and Debugging:**

Once everything was functionally sound, the team gathered to discuss how to approach system testing. It was clear that everyone would need to pick carefully through each scenario to ensure that there were no major bugs within the code.

However, to increase team efficiency, the group broke up again to tackle different major projects. Henry remained on the testing side by running various scenarios to test the internals and GUI elements of the game. David focused on creating some of the mentioned non-functional features. Lastly, George acted as a middleman, providing support for David when running into feature issues as well as testing the game himself to help Henry identify any small bugs.

Lastly, the game was release ready from the multitude of user acceptance tests from friends and family. Overall, the final project released, though simple, is creative and robust.

**GitHub Repository Link (Source Code):**

https://github.com/Gonzaga-CPSC-224/cpsc224\_fp\_f22-lilyahtzee

# Future Work

The future of this project could take a multitude of different directions. Though Lil Yahtzee accomplished all of the functional requirements provided, there is ample space for enhancement with the game itself.

One potential path would be to improve the visual aspects even more. One consistent issue with user acceptance testing was attempting to explain how exactly Yahtzee is played and how one can win. A simple fix would be to provide an extensive instructions screen before the game begins. Another visual aspect that could use some improvement is the scorecard. As of right now, the dynamic scoring is held in a text field that is very simple and difficult to read. Research into potential Swing tables or visual formatting to make the scorecard look cleaner would be a major feature to add.

Another potential path to take the future of this project would be to research ways to package and release the game as a program to anyone. Rather than having to run the game through VSCode or the terminal, creating a way to release a download that could run the program with the click of a button would be another fantastic way to take this project to the next level.

# Glossary

**Object-Oriented Programming:** Computer Science principle for organizing program code that includes using simple things to represent complexity (abstraction), combining methods/properties related to the same object (encapsulation), extending implementation of existing objects (inheritance), and creating objects of different types (polymorphism).

**Model View Controller:** Computer Science principle used to develop interfaces by breaking up the application into three interconnected parts: model (manages data and app rules), view (output representation), controller (accepts input/converts to command).

**Unit Testing:** Unit tests are used to test a piece of software part-by-part. Rather than testing a whole class or program to catch a bug, specific functions that are essential to the program are checked with test data to see if it remains within its bounds and acts accordingly when given a case on the edge of these bounds. In VS Code, JUnit Tests are utilized to perform these tests.

**Integration Testing:** Integration tests check the interactions between small components of a program. This is the next step taken after unit tests to find bugs that passed the unit tests but may have not necessarily been checked in its interaction with another function.

**System Testing:** System testing is the umbrella name for all types of tests regarding how the components, when put together, react in the program’s entirety. The program is checked to identify failures to reach the project’s original functional requirements.

**Functional Testing:** Using the requirements specifications provided by the project manager, functional testing checks that all the specifications are working properly and have been implemented correctly.

**Performance Testing:** Checks the goals of the team that go beyond the project’s initial specifications. This includes testing the program for inputs that exceed expected inputs, proper layout designs, and program effectiveness. It identifies “how the stability, speed, scalability, and responsiveness of an application holds up under a given workload” (“What is Performance Testing?”).

**User Acceptance Testing:** This is used to assure that the program is tailored correctly for the user. Furthermore, it ensures that any overlooked tests are identified by outside groups that will ultimately be using the program. This is the last confirmation required before a project can be sent out for final release.

# References

“The History of Yahtzee.” *UltraBoardGames*, https://www.ultraboardgames.com/yahtzee/history.php.

“What Is Performance Testing?” Micro Focus. Accessed November 17, 2022. https://www.microfocus.com/en-us/what-is/performance-testing.

“Yahtzee.” *Wikipedia*, Wikimedia Foundation, 24 Nov. 2022, https://en.wikipedia.org/wiki/Yahtzee#:~:text=Yahtzee%20is%20a%20dice%20game,Edwin%20S.%20Lowe%20in%201956.

# Appendices

**Appendix A: Project Initial Plan Designs**

**Text

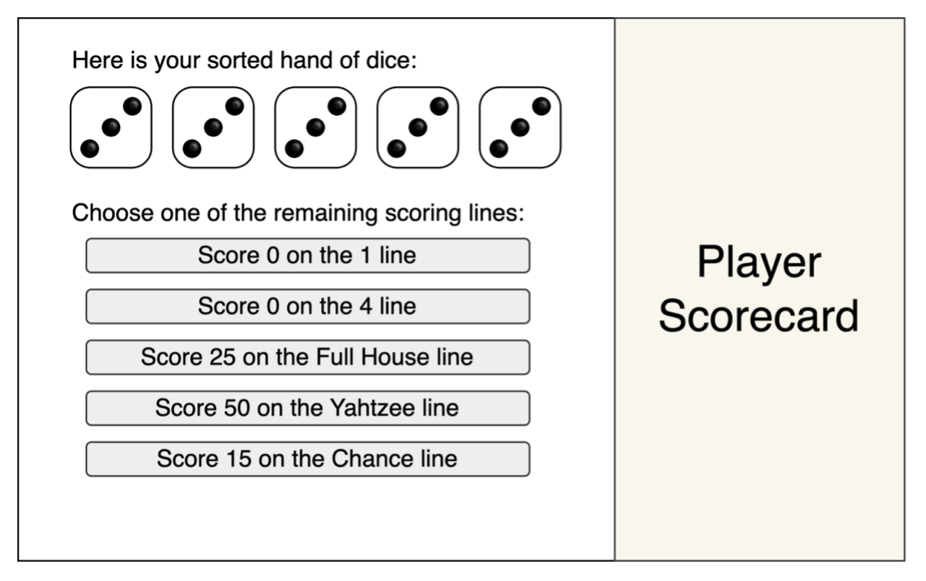
Description automatically generated**

**Text

Description automatically generated**

**Diagram

Description automatically generated**

****

**Text

Description automatically generated**

**Appendix B: Project Final Program Designs**

**Graphical user interface, text, application, website

Description automatically generated**

**Graphical user interface, text

Description automatically generated with medium confidence**

**Text

Description automatically generated**

**Graphical user interface, application

Description automatically generated with medium confidence**

**Table

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**Graphical user interface, table

Description automatically generated with medium confidence**

**Graphical user interface, application

Description automatically generated**