Software Requirements Specification

for

Hound Army Checkers

Version 1.5 approved

Prepared by Samantha Coyle, Spencer Albert, Taylor Moralez, and Rowan Stone

CS3398 Fall 2018: Software Engineering, Section 001

September 12, 2018

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

3. External Interface Requirements 3

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

4. System Features 4

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

5. Other Nonfunctional Requirements 4

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

6. Other Requirements 5

Appendix A: Glossary 5

Appendix B: Analysis Models 5

Appendix C: To Be Determined List 6

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Taylor Moralez | 10-23-18 | Changed extra features from king mode and avatar selection to shot clock and hint square. | 1.01 |
| Samantha Coyle | 10-24-18 | Added to the User Classes and corrected bulleting. | 1.02 |

|  |  |  |  |
| --- | --- | --- | --- |
| Samantha Coyle | 10-27-18 | Added more pictures to clarify points and updated additional feature requirements. | 1.03 |

|  |  |  |  |
| --- | --- | --- | --- |
| Team | 10-28-18 | Finalized document | 1.04 |

|  |  |  |  |
| --- | --- | --- | --- |
| Samantha Coyle | 10-29-18 | Added test button. | 1.05 |

# Introduction

## Purpose

The purpose of this document is to provide a detailed description of the requirements for the Hound Army Checker Board game. This document will cover the details on the functionality, design specifications, features, and requirements for the game. Hound Army shall provide the user a functional, high-quality checkers game. This document is version 1.5. The SRS scope will include a full description of the requirements and functional deliverables that the Hound Army shall produce.

## Document Conventions

The naming standards included within this document are as stated below. The acronym is on the left and the full-description of the acronym is on the right-hand side. Priorities for higher-level requirements are assumed to be inherited by detailed requirements.

|  |  |
| --- | --- |
| **Acronym** | ***Definition*** |
| HAC | *Hound Army Checkers* |
| *GUI* | *Graphical User Interface.* |
| *AI* | *Artificial Intelligence* |
| *GE* | *Game Engine* |

## Intended Audience and Reading Suggestions

This document is intended for developers so that they may follow the requirements listed within the SRS as they work on developing the product. It is intended for project managers so that they may ensure that the requirements are met. The marketing staff will utilize this document to boast its features, functionality, and efficiency to the users of the checker board game. Testers will also use this document, following it in sequential order, to ensure that all requirements are met, and product has no issues. The rest of the SRS will contain an overall description of the Hound Army Checker Board game, list the external interface requirements, system features, nonfunctional requirements, and other requirements.

Additionally, there is an appendix section at the end of the requirements section to disclose a glossary and analysis models. This document is organized with a general description, followed by a more detailed list of requirements. It is suggested to read this document in sequential order starting at the beginning, until the end of the document is reached so that the reader may get a good grasp of the goals, overview, and features of the game. It is recommended that all readers start at the beginning to get an understanding of the game and proceed from there. Developers, project managers, and testers should read the entire document to fully understand the product. The marketing staff does not need to understand the entirety of the requirements section but may want to read through it to ensure full comprehension of the game.

## Product Scope

Hound Army shall provide a fair checker board game utilizing JavaScript. The benefits of this product include a traditional checker board game for the entertainment and enjoyment of the user. Benefits include improving ones’ checker board game skills.

## References

“Checkers: American.” *ItsYourTurn.com - Help Page*, It's Your Turn, Inc, 2017,

[www.itsyourturn.com/t\_helptopic2030.html](http://www.itsyourturn.com/t_helptopic2030.html).

# Overall Description

## Product Perspective

HAC is a stand-alone application that will serve as a digital version of the American iteration of the classic board game, Checkers. Checkers is a turn-based, strategy game that requires two players. The American version of the game is played on an 8 x 8 checkerboard, with the starting number of 12 uniformly shaped game pieces per player, termed “men”. One player will use light pieces, and the other will use dark pieces. Player pieces are only allowed to occupy the dark squares on the checkerboard. The player with the dark pieces will move first. The objective of Checkers is for the player to capture all their opponent’s pieces, or to otherwise prevent them from being able to make a legal move.

As far as moves go, legal moves consist of either “simple “moves, where the player moves their piece to an adjacent, diagonal space going towards the opponent’s side of the board; “jumps”, where the player will “jump-over” a single opponent’s piece that is on an adjacent, diagonal space to occupy the next empty space in the same direction, and “multiple jumps”, where the player performs at least two “jumps” in a row when it is valid for the player to jump over another piece after the first jump has been made, either in the same diagonal direction or a different one. When a player’s “man” reaches the row closest to their opponent on the board, known as “king’s row”, the piece is upgraded to the status of “king”. Kings can move in any diagonal direction, forward or backward.

## Product Functions

* HAC shall allow the user to reference the player manual during the game.
* HAC shall allow the user to turn music/sound on or off at will.
* HAC shall display a shot clock for the players during the game.
* HAC shall display whose turn it is during the game.
* HAC shall only allow each player to move their own pieces during their turn.
* HAC shall highlight optional moves for a piece when selected.
* HAC shall allow the user to move a piece to their selected destination as long as the move is valid.
* HAC shall not move a piece for the user if the user’s chosen move for that piece is invalid.
* HAC shall provide an accurate and up-to-date score on the scoreboard.
* HAC shall remove captured pieces from the board.
* HAC shall king a player’s piece as soon as their piece reaches king’s row on their opponent’s side.
* HAC shall notify the user when the game has ended and then stop gameplay.
* HAC shall notify the user if they have won or lost the game.
* HAC shall allow the user to test the functionality of the game if desired.

## User Classes and Characteristics

There will be no separate user classes, as each user will have the same experience and options when playing HAC. There are no special privileges or additional expertise required to play. The user is not expected to know how to play checkers, as the rules will be depicted on the UI.

## Operating Environment

The software will operate on a modern web application environment, which will be accessible on Google Chrome or Mozilla Firefox. One of these browsers is needed in order to play the game.

## Design and Implementation Constraints

HAC will only be available in English. The software is expected to be completed by October 29, 2018. HAC will not save or ask for any personal user information; therefore, there will be minimal security considerations. In order to play the game, the user must have either Google Chrome or Mozilla Fox to run the application. Audio capabilities and/or speakers are required in order to take advantage of the music capabilities of the application.

## User Documentation

A short user manual will be included as a text document to provide users with an overview of how to operate HAC. A Checkers How-To-Play reference guide will be included to provide the general rules for the game.

## Assumptions and Dependencies

An AI module, aptly named ‘Player 2,’ will be created and utilized for the computer opponent that will play against the user. It will be assumed that all music and/or sound effects used for the game will be under a creative-commons use license. Graphics and photos to be sourced will be created specifically for use in HAC. It will be assumed that the testing of the game will be done prior to the actual game play.

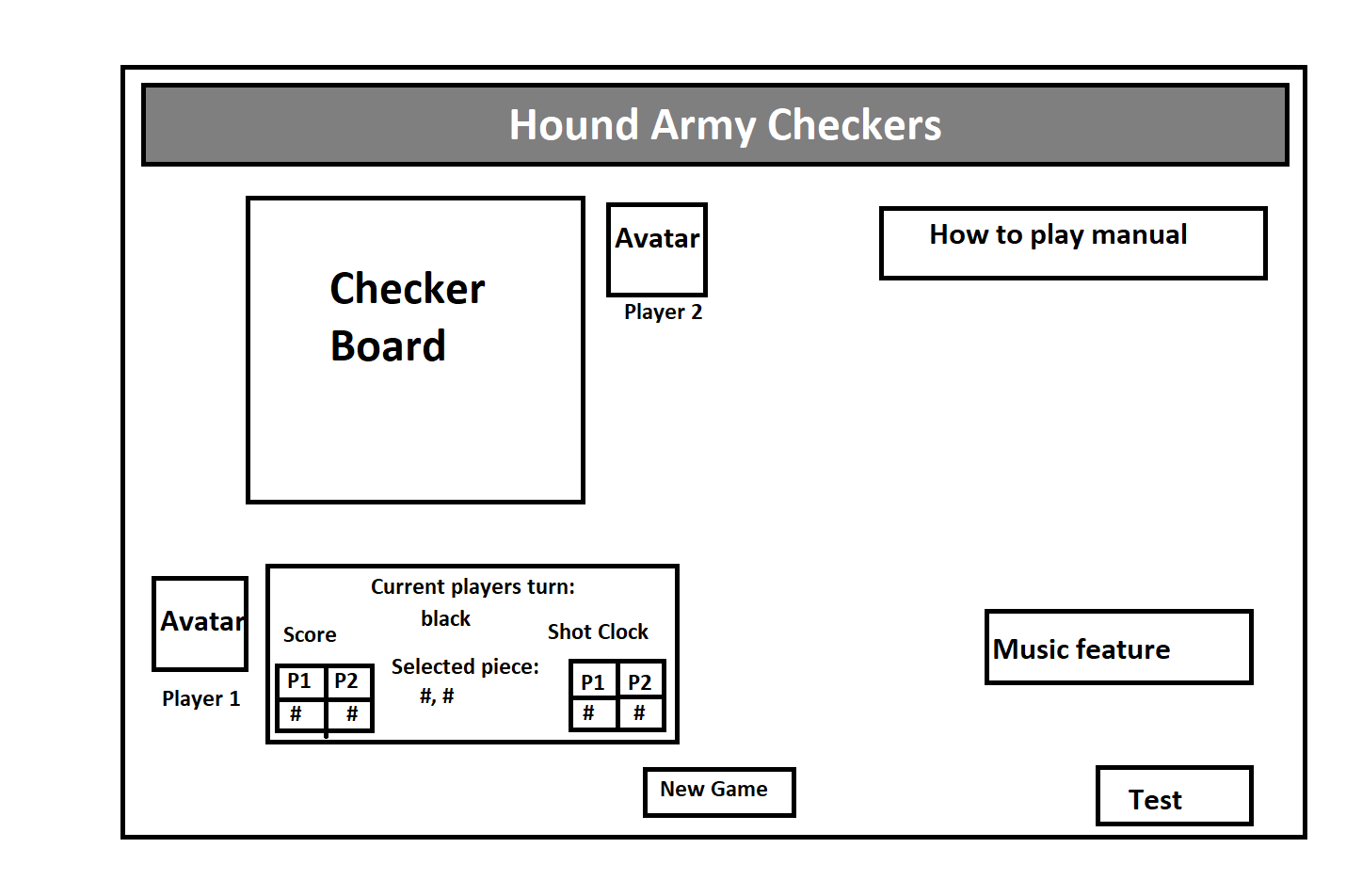
# External Interface Requirements

## User Interfaces

The game shall be interacted with through a single screen, on which the user will find the game board towards the top-left of the screen. This is where the user will interact with their pieces during the game, as well as configure options for starting a new game. Below the game board area will lie a variety of information such as the current score of the game being played, options such to turn on the music feature, as well as a ‘New Game’ button, which will restart the game. There will also be a shot clock to display the current time each player has consumed totally for their game time, as well as whose current turn it is, and information on the current piece selected. There are also avatar images for each player for them to better distinguish from one another. On the right-hand side of the screen will be a user manual menu with information about the game, including the rules for checkers. There shall also be a test button to perform the testing of the checker board functionality.

During gameplay, the user will move their pieces by first clicking on one of their pieces, and then clicking on the space that the user would like to move their piece to.

Below is an approximate example of the layout of the program.



## Hardware Interfaces

There will be no additional hardware requirements besides a mouse and keyboard that is compatible with the user’s computer and operating system environment. In order to take advantage of the music feature, audio capabilities and/or speakers are required for the application.

## Software Interfaces

HAC will be a standalone application, requiring only a compatible web browser to run (either Google Chrome or Mozilla Fox.

## Communications Interfaces

HAC will be a standalone application, with no communication with any other programs or in-between different users.

# System Features

## Game Screen

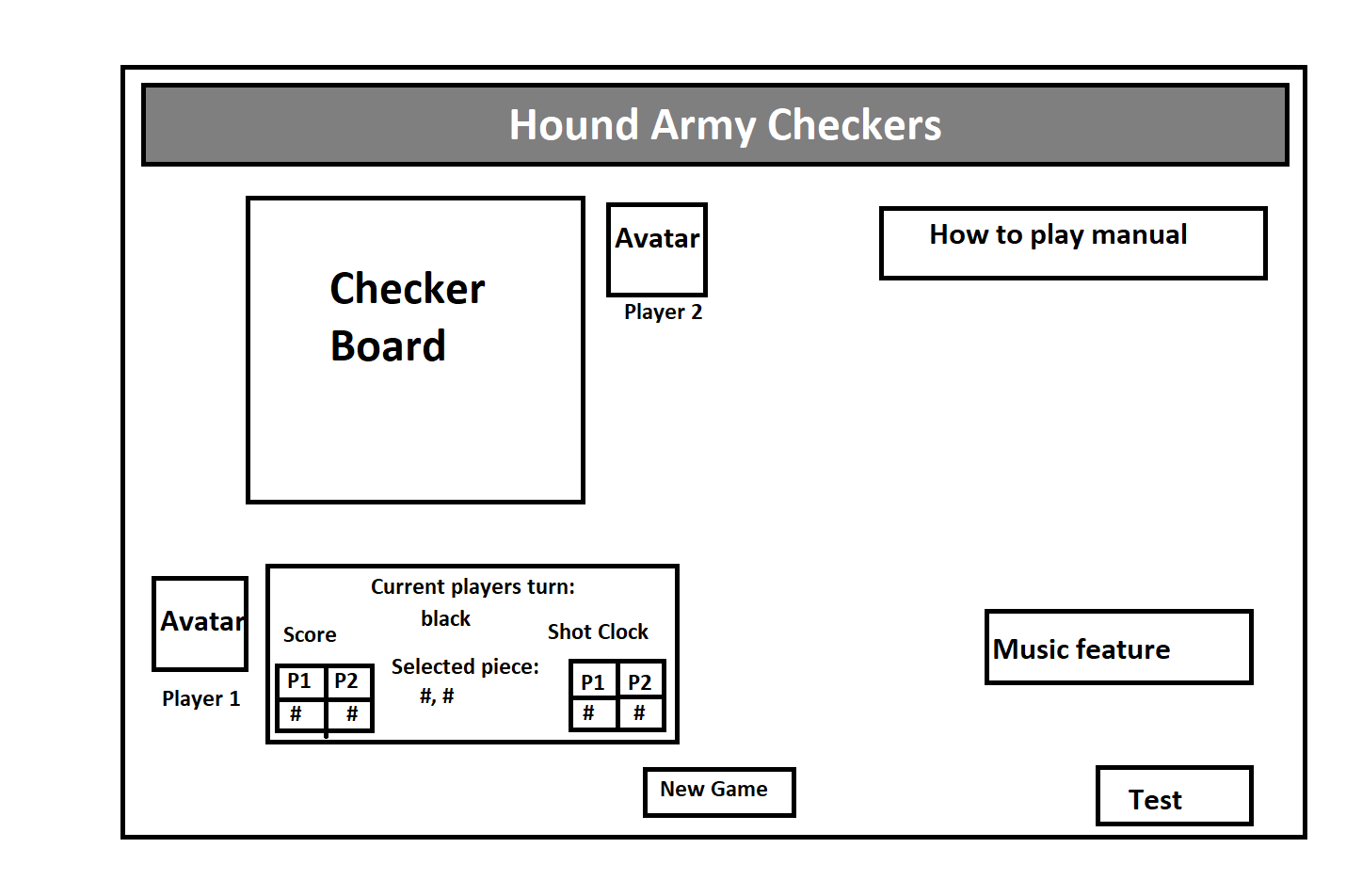
4.1.1 Description and Priority

The overall user interface shall include the checkers board game, a text area depicting whose turn it is in the game, the coordinates of the piece selected, a shot clock with the total amount of time that each player has consumed during the game, and a tally of the scoreboard for each player. Also, toward the end of the UI shall include a play/pause button for the music for during the game, as well as a button to click to start the game over. The interface shall have a menu option for users to select to read more in-depth information on directions and features of the game. This feature is described in further detail in section 4.4. There shall be a test button at the bottom right of the UI to test the selection of a piece, as well as the highlighting of optional moves.

The aspects of the user interface are of high priority as they benefit the user in understanding how to play the game and exploit the different features of the game. The benefit factor is 9 as this is essential to playing the game, it is penalty factor is 0, and the risk factor is also 0.

4.1.2 Stimulus/Response Sequences

* If the user opens the web application, then the checker board game, additional feature toggle buttons, scoreboard, shot clock, text field for whose turn it is, test button, and the user manual shall appear:



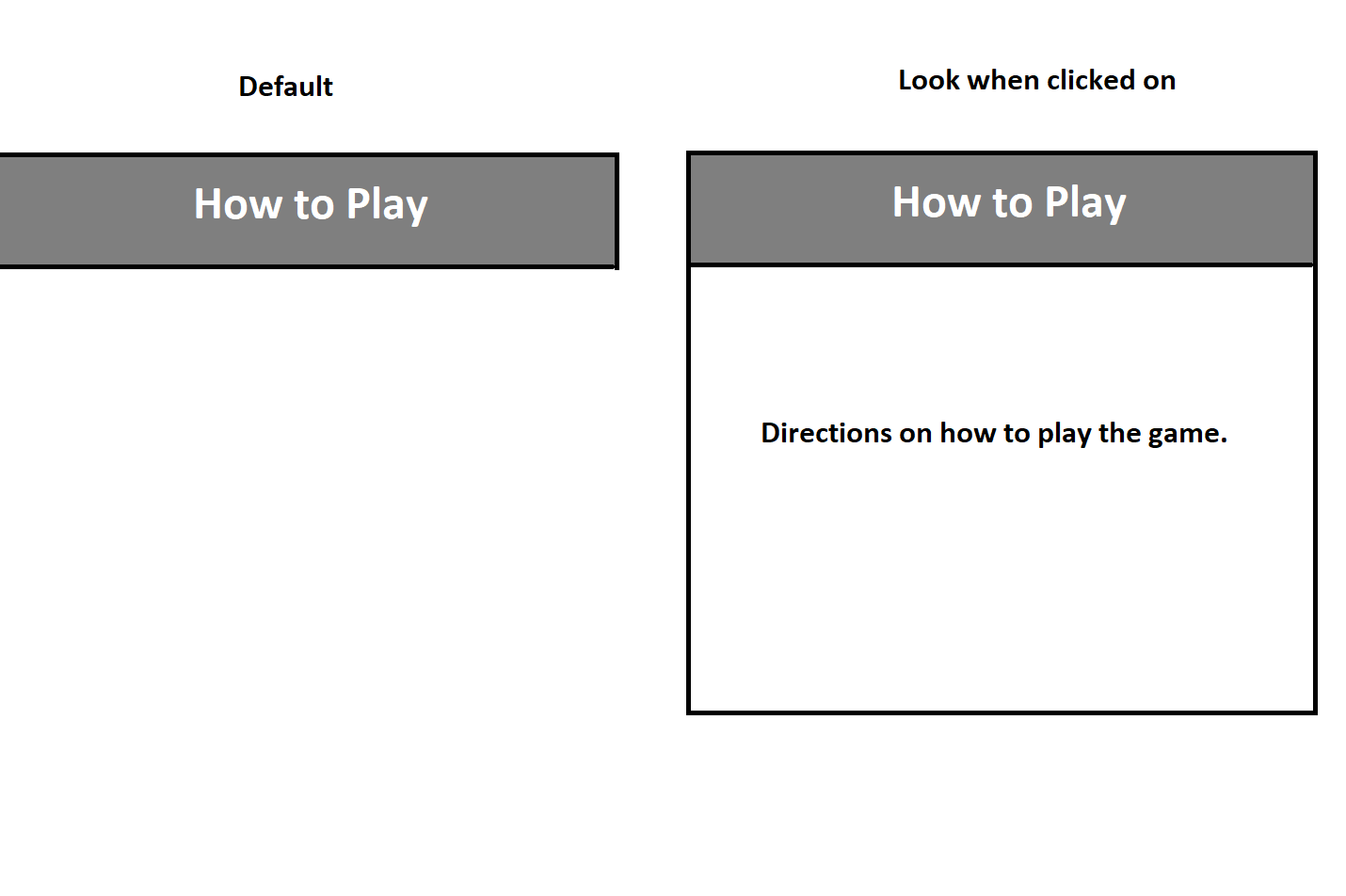
* When a player wins the checker game, an alert shall be displayed depicting the game winner.
* If the start over game button is clicked, the checker game board is cleared and refreshed to its initial condition.

4.1.3 Functional Requirements

REQ-1: If the user’s laptop/PC has visual capabilities, then the game screen shall load up properly without error.

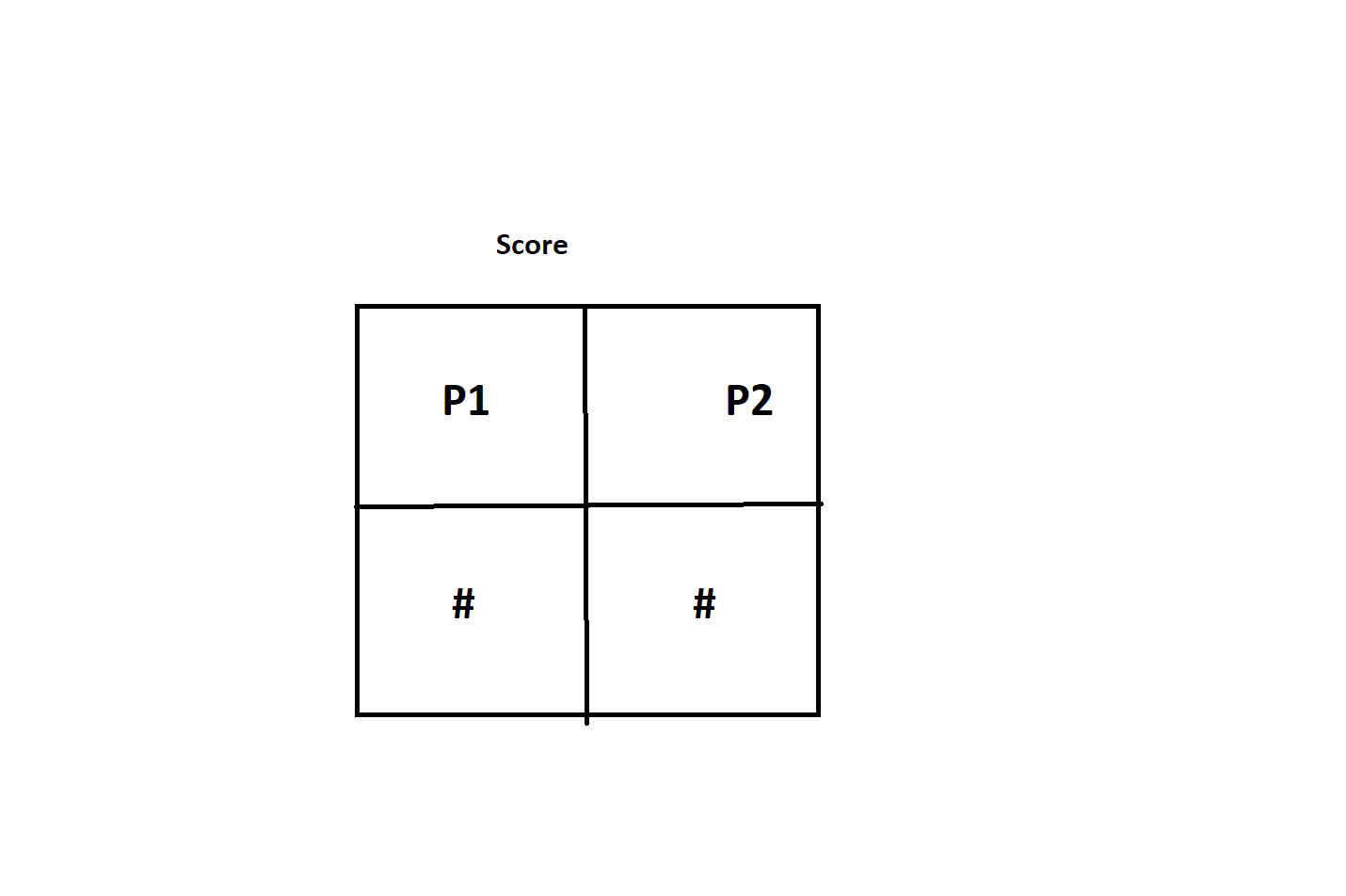
REQ-2: The game screen shall display the checkers game board, user manual, score board, shot clock, text area to depict whose turn it is, text area to display the selected piece coordinates, scoreboard, music play/pause button, and a start over button. These features will be functional.

REQ-3: The game screen user manual shall be clickable to drop down information on how to play the game to the user if selected.



REQ-4: The game board itself shall be interactive and allow the user to move their game pieces to valid spaces.

REQ-5: The scoreboard shall maintain an accurate account of the number of pieces that each player has jumped over through the score board. Players shall be denoted by P1 and P2, indicating player 1 and player two.



REQ-6: The music play/pause button shall have the capability to play music or pause the music. It shall also have muting capability.



REQ-7: The start over button shall start the game over if clicked.

REQ-8: The game screen shall alert the user when the game is over who won overall.

REQ-9: The test button shall alert the user that the test case is in progress once clicked on.

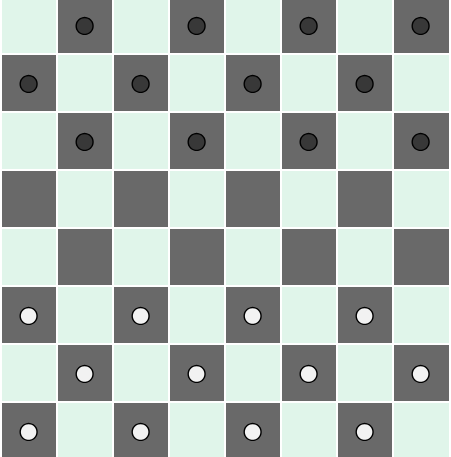
## Game Engine

4.2.1 Description and Priority

HAC will feature a game engine that is responsible for initializing the board state at the beginning of each game, determining the validity of any moves made by the user, making the moves for the AI opponent, and determining when a game has ended, and which player has won. This is a high priority feature, as it is responsible for the basic functionality of the game.

4.2.2 Stimulus/Response Sequences

* If the user opens the application, then HAC shall display the checker game board with the proper checker game pieces so that the users may start the game.



* If the user selects to move a game piece, then HAC shall allow the user to click where they want to place their piece. If the user decides to move the game piece to an invalid space, then HAC shall notify the user of the invalid move. If a user has tried to move to an invalid space, HAC shall require a reselection of a piece to move.
* Once the user has completed their turn with a valid move, HAC shall choose a valid move for the AI opponent.
* If the player gets a piece to the opponent’s end of the board, HAC shall make that piece a king piece.
* When the game is over, and one player has no more game pieces to play or if they can’t make a move, then HAC shall notify the user who has won the game.
* If the user can jump over multiple opponent pieces, then HAC shall enforce the user to jump as far as possible until there are no more available jumps for that piece.

4.2.3 Functional Requirements

REQ-1: If the user clicks on a game piece that is their own, then the game board shall allow the user to move the game piece to a valid space.

REQ-2: If the user’s turn is up, then HAC shall have the AI opponent complete their turn.

REQ-3: If the game is over, then HAC shall notify the user who has won the game.

REQ-4: If the user tries to move the game piece to a non-valid space, then HAC shall notify the user that the move is invalid and shall not move any pieces. HAC shall require a reselection of a game piece in order to move a piece to after an invalid destination is chosen.

REQ-5: If the user tries to move a game piece that is not their color, then HAC shall notify the user that they are the other color game pieces and no pieces shall be moved.

REQ-6: If the user starts a new game, then HAC will initialize the board with all the pieces, and HAC will reset the score and shot clock of both the player and AI opponent.

REQ-7: If the user selects a game piece, then HAC shall allow the user to click where they want to place the piece.

REQ-8: HAC shall keep track of the number of game pieces that a player has left to determine if the game is won/lost if one player has no more pieces, or if they can no longer make a move.

REQ-9: When a piece gets to the end of the board, HAC shall make that piece a king.

REQ-10: HAC shall keep track of the number of available jumps for the game piece that the user selects to ensure that the user makes a valid move and moves their chosen piece as far as possible.

REQ-11: HAC shall keep track of a game piece status (king or regular), so that pieces that are already kings cannot become kings again.

## System Sound

4.3.1 Description and Priority

The game shall provide the player with the option to turn music on, off, or on mute during the game. This feature will be implemented to provide a more enjoyable gameplay experience for the user as well as additional gameplay feedback that users will find satisfying and more immersive. This is a low priority feature.

4.3.2 Stimulus/Response Sequences

* If the user clicks the play button once while HAC is running, the game shall play sounds when pieces are moved.
* If the user clicks the sound button a second time while HAC is running, the game shall stop playing sound effects when pieces are moved.
* If the user clicks the volume button once while HAC is running, the game shall mute the music.
* If the user clicks the volume button a second time while HAC is running, the game shall unmute the music.

4.3.3 Functional Requirements

REQ-1: If the user’s computer has a functional sound card and audio capabilities, then the sound and music play/pause/mute options shall be available for the user to use.

REQ-2: If the user selects the play button, then HAC shall play music.

REQ-3: If the user selects the play button a second time, then HAC shall pause the music.

REQ-4: If the user selects the volume button, then HAC shall mute the music.

## In-Game Manual

4.4.1 Description and Priority

The in-game user manual shall be displayed on the right side of the game screen. It shall display to the user a drop-down menu option of different features and rules in detail so that they can have a better understanding of some of game and its additional features.

This is of medium priority as it is not essential to have, but it will aid in the user experience and understanding of the application. It is given a score of 7 for benefit to the user, 0 for penalty as it does not harm the experience of the user, and it will be a risk factor of 0.

4.4.2 Stimulus/Response Sequences

* If a drop down on the in-game manual is clicked, more information on chosen subject shall appear.
* If the drop-down on the in-game manual is clicked a second time, the additional information shall hide again until re-clicked.

4.4.3 Functional Requirements

REQ-1: If the game screen is opened, all drop-downs in the in-game manual shall be kept closed until opened by user.

REQ-2: If the game screen in-game manual drop-down title button is clicked, then HAC shall display the appropriate information to the user.

REQ-3: HAC shall allow an in-game manual drop-down to be closed if clicked a second time.

## Scoreboard

4.5.1Description and Priority

The game screen shall display a scoreboard underneath the gameboard, with the display showing the user how many pieces they have taken from the opponent, and how many pieces the opponent has taken from the user. This feature is of medium priority.

4.5.2Stimulus/Response Sequences

* If a user starts a new game, then scoreboard shall be reset to 0.
* If a user takes an opponent's piece, or if a user has a piece taken, then the scoreboard shall be updated.
* If a game ends, then the scoreboard shall be reset to 0.

4.5.3Functional Requirements

REQ-1: When a checker piece is taken off the board, then the scoreboard shall be updated correctly to whoever took the piece (user, or opponent) before the next turn takes place.

REQ-2: When a game ends or a new game begins, the scoreboard entries for both players shall zero out.

## Shot Clock

4.6.1Description and Priority

The game shall have two separate timers that will determine the time elapsed (in seconds) during the turns of both the player and AI, similar to that of the clocks used in chess (but will be counting up rather than down). This is a low priority feature, as its inclusion is not integral to the game of checkers but is rather an extra feature.

4.6.2 Stimulus/Response Sequences

* At the start of a new game, both timers shall be set to 0.
* When it is the player’s turn to move, then the shot clock shall start counting upwards, and stop once the player has finished their turn.
* Once the player has finished their turn and it is the AI’s turn, the AI clock shall start counting upwards, until it has made its move.
* When it is the player’s turn again, the shot clock shall continue counting upwards from the last number that it reached when it last stopped. It shall continue counting until the player’s turn is ended, then the AI clock shall continue counting upwards.
* This will continue until the game is over, and both clocks shall stop counting.

4.6.3Functional Requirements

REQ-1: The game screen shall have a small table underneath the game board labeled ‘Shot Clock,’ with two columns labeled ‘P1’ and ‘P2,’ each with an integer value beneath it to represent the time consumed.

REQ-2: The HAC shot clock numbers shall count upwards, in seconds, starting from 0, each only incrementing during the respective player’s turn.

REQ-3: At the start of a new game, the shot clock timers shall reset to 0.

## Hint Squares

4.7.1 Description and Priority

When a player clicks on the piece that they wish to move, the possible squares that they can move to will be highlighted, in order to aid the player. This is a low priority feature.

4.7.2 Stimulus/Response Sequences

* When it is the player’s turn, and they click on the piece they wish to move, the available squares that they can move to will be highlighted.
* Once the player has either moved or selected a different piece, the squares will cease to be highlighted. If the player has selected a different piece to move, the available moves for the new piece will then be highlighted.

4.7.3Functional Requirements

REQ-1: When a piece is selected, the available valid destination squares shall be highlighted in yellow.

REQ-2: When a piece is moved, the highlighted squares shall be un-highlighted.

REQ-3: If a piece is selected, then deselected, the highlighted squares shall be un-highlighted.

# Other Nonfunctional Requirements

## Performance Requirements

Hound Army Checkers shall not crash or freeze for the user. Beyond that, there have not been any specific performance requirements set. The overall goal of Hound Army Checkers shall be to get the program running smoothly for the user, without any bugs or other user related problems.

## Safety Requirements

There are no safety requirements for this project.

## Security Requirements

The user will not be inputting any sensitive information, and none will be kept by Hound Army Checkers, so there are no security requirements.

## Software Quality Attributes

Hound Army Checkers shall run well, without bugs, and give the user an experience in comparison to other online checker games with similar features.

## Business Rules

There are no business rules for Hound Army Checkers.

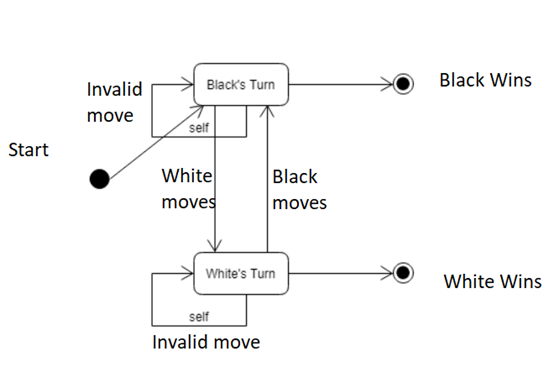
# Other Requirements

Appendix A: Glossary

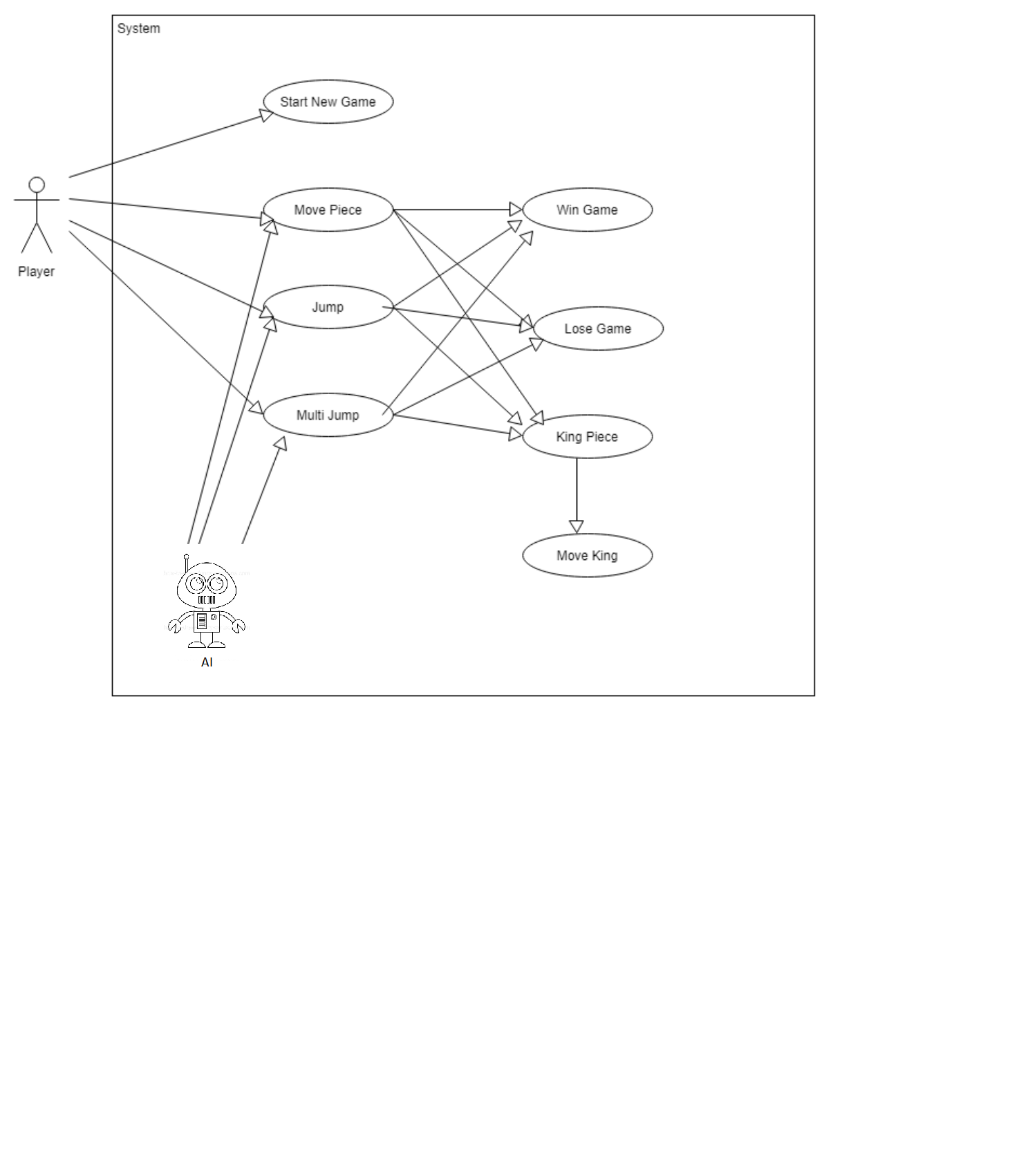
|  |  |
| --- | --- |
| HAC | Hound Army Checkers |
| AI | Artificial Intelligence |

Appendix B: Analysis Models

The diagram below is a UML state machine diagram to represent the different states that the game could be in. It will either be black’s turn or white’s turn, and the state will alternate between the two until one player wins. A player can win the game by either jumping over all of their opponents' pieces, or if the opponent has no valid moves.



The diagram below is a UML use case diagram. This diagram displays the behavior of the system and the actions therein. It shows the actions that the player may have, and the interaction of the system to those actions. The AI is represented as a sort of actor within the system as it is implemented within the system, but will have almost the same behaviors with the system as a regular player would, minus the ability to start a new game.



Appendix C: To Be Determined List