# Software Requirements Specification

# Hanabi Client G3

Authored by:
Joel Siemens <jms676>
Olivier Lacoste-Bouchet <oll096>
Troy Belsher <twb431>
Kole Barber <klb731>
Destin Astrope <dja392>

February 3, 2019

# 1. Introduction

## 1.1 Purpose

The purpose of this document is to provide in-depth information about a software version of the cooperative card game Hanabi. This document will outline the features, interfaces, and constraints of the software.

## 1.2 Scope

This software version of Hanabi will be playable for two to five players at once in a real time environment. The players will cooperate through hints to complete fireworks. The game software will include artificial intelligence (AI) to support computer players along with human players at the same time. The game will be playable online through a dedicated server.

## 1.3 Definitions, Acronyms, and Abbreviations

A number of abbreviations and acronyms will be used in this document to explain concepts. These definitions will help the reader to understand some of these definitions in the context of this document.

“UofS” refers to the University of Saskatchewan.

"Player", for the purposes of this document, will refer to the human or AI user of the program.

"Client", by contrast, will refer to an executing program, typically in relation to the server to which the program is designed to connect.

“AI” refers to artificial intelligence, a system by which the game will be able to “play itself” without input from a human player.

“GUI” refers to Graphic User Interface, which is the visual aspects of the program shown to a user during the program’s use.

“Spinks” refers to a building on the UofS campus. There are several computers available for students.

## 1.4 References

Hanabi is the card game which is being referred to in this document.  It is assumed that the reader will know how the game itself is played, but a copy of the game rules can be found here: http://www.boardgamecapital.com/game_rules/hanabi.pdf

## 1.5 Overview

This document will begin by describing the program and its requirements, and the general use of the functions that will contribute to the use of the program.  The following section will go into more detail about these functions, as well as the user interface requirements. The document will then discuss some of the requirements and limitations of the software.

# 2. Overall Description

## 2.1 Product Perspective

This Hanabi software application is developed for everyone who wants to play the cooperative card game Hanabi either with friends or with computer players. The application is an independent game client that can be run on a user's local machine to host or join games of Hanabi on a separately constructed server located on the University of Saskatchewan's network. These games can be populated and played by users running this client version of Hanabi.

### 2.1.1 System Interfaces

There are two parts to the software version of Hanabi:

- User Client – Establishes a connection to the game server, continuously collects and analyzes individuals' game data and displays it appropriately to the local user.
- Server – Transmits details of player actions to the necessary users in-game.

### 2.1.2 User Interfaces

The software interacts with the user using a Graphic User Interface. This GUI displays the menu, game and other screens to the user and allows them to perform all actions through a graphical environment using a keyboard and mouse. See Figures 3 - 15 located in section 3.1.1.

The AI portion of the software will be executed and controlled solely by a command-line interface. However, other human users in the same game as the AI will be able to see a graphical representation of the AI. In game an AI player will appear the same as a human player.

### 2.1.3 Communications Interfaces

The software requires TCP in order to communicate with the Hanabi server.

### 2.1.4 Operations

There are four operation modes within the system:

- Menu: The menu requires user input to navigate the menu, as well as start a game lobby.
- Waiting for players: This operation requires no user input. This operation continues until the proper amount of players have connected.
- In game, client player's turn: The game requires the player to choose a move. This operation will last until the previously selected time limit expires.
- In game, idle: This operation requires no input from the user, as they must wait for others players to play. Players can choose to interact but can make no impact on the current game state.

## 2.2 Product Functions

The software performs the following functions. Each function's availability is dependent on the current state of the system.

### 2.2.1 Host a game – Available on main menu

This function allows the user to create a new game of Hanabi on the server. This requires the user to enter a turn time limit, their NSID and the total number of players going to play. After creation of the game, the user will join automatically. The user is given a unique signature key that is used by others to join the new game. The game will start automatically after the set number of players join. See 3.2.1 for function details.

### 2.2.2 Join a game – Available on main menu

This function allows the user to join an existing game of Hanabi on the server. This requires the user to enter a signature key to an already existing game. After joining the game, the user will wait for the game to start. The game will start automatically after the set number of players join. See 3.2.2 for function details.

### 2.2.3 How to play – Available on main menu

This function allows the user to read the Hanabi game rules, instructions on how to play and help with using the GUI. This requires the user to click the "How to play" button. After reading the user can close the window. See 3.2.3 for function details.

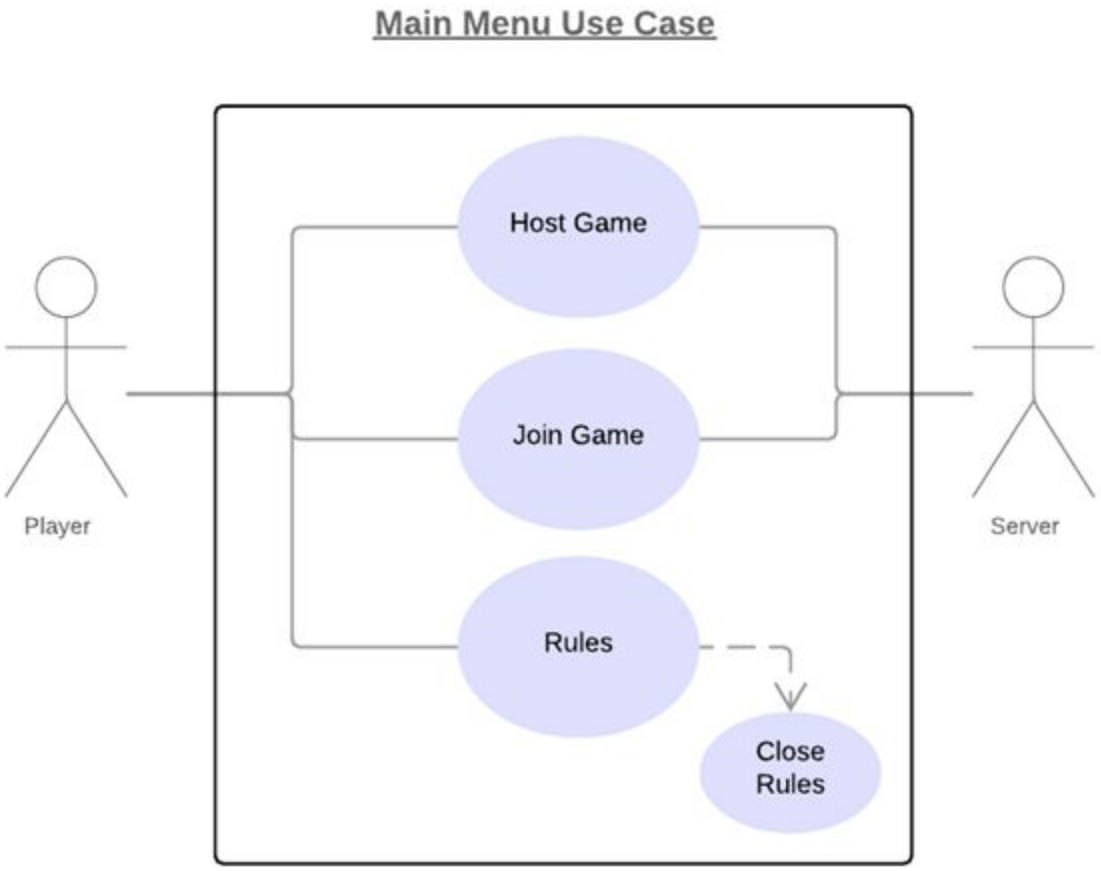


Figure 1

### 2.2.4 Play a card – Available in-game as the active player

This function allows the user to play a card in-game. This requires the user to click the “Play a card” button and then select a card from their hand. After playing the card of their choice, they are dealt another card (if applicable) and their turn is ended. See 3.2.4 for function details.

### 2.2.5 Discard a card – Available in-game as the active player

This function allows the user to discard a card in-game. This requires the user to click the “Discard a card” button and then select a card from their hand. After discarding the card of their choice, an information token is returned to play, they are dealt another card (if applicable) and their turn is ended. See 3.2.5 for function details.

### 2.2.6 Give colour info – Available in-game as the active player

This function allows the user to give information about one and only one colour to another player in the game. This requires the user to click the “Give colour info” button and select a card with the colour to be shared from the hand of the player who will be receiving the information. After giving the colour information, an information token is removed from play and the user’s turn is ended. A colour info marker is added to the clued card. See 3.2.6 for function details.

### 2.2.7 Give number info – Available in-game as the active player

This function allows the user to give information about one and only one number to another player in the game. This requires the user to click the “Give number info” button and select a card with the number to be shared from the hand of the player who will be receiving the information. After giving the number information, an information token is removed from play and the user’s turn is ended. A number info marker is added to the clued card. See 3.2.7 for function details.

### 2.2.8 Look at discard pile – Available in-game

This function allows the user to look at the discard pile anytime during in-game play. This requires the user to click the “Discard Pile” button. After the user is done looking at the discard pile, the window can be closed by click the “Close” button. See 3.2.8 for function details.

### 2.2.9 Exit – Available in all windows

This function allows the user to exit the software at any time. This requires the user to click the “Exit” or “Close” button in the top right corner of the client. After exiting the software, the program window is closed.  See 3.2.9 and 3.2.10 for function details

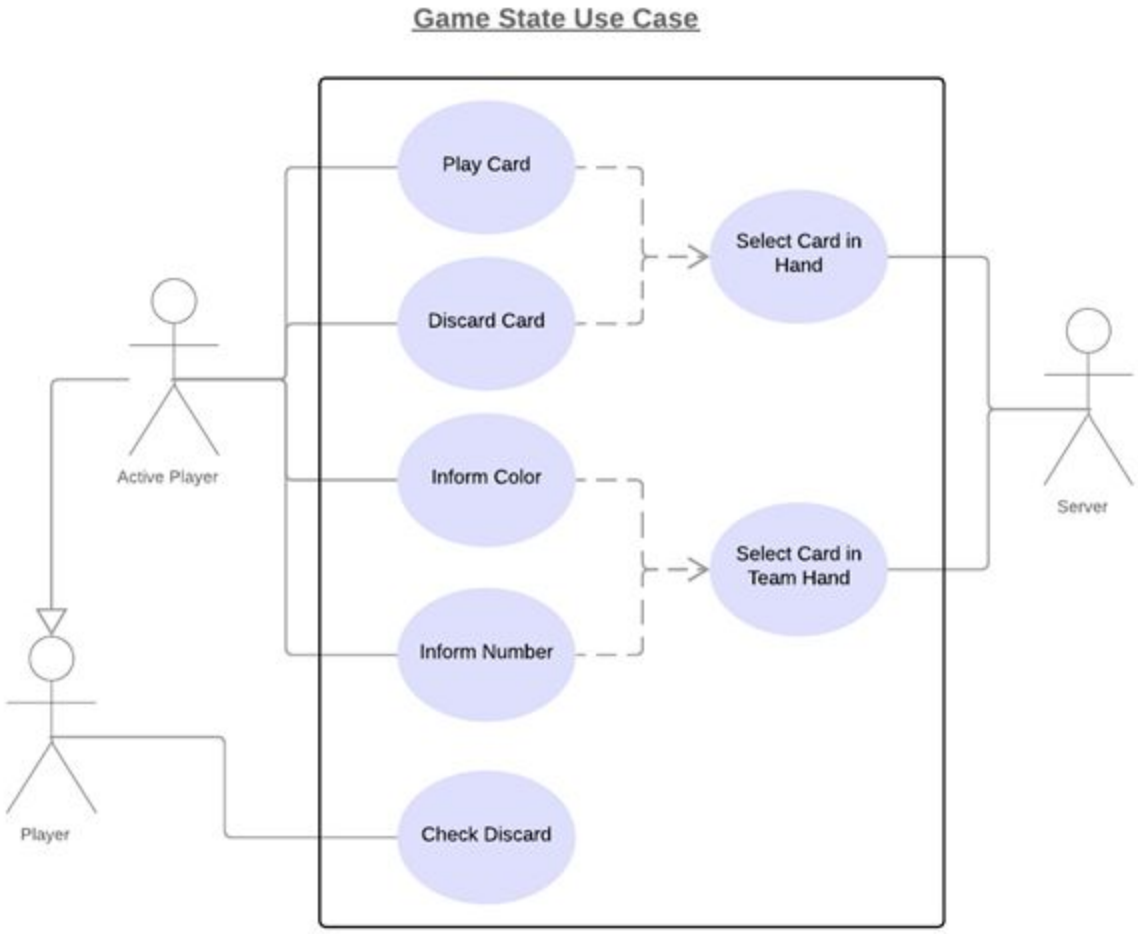


Figure 2

## 2.3 User Characteristics

To play a game of Hanabi using our system, a person must be able to read and comprehend English, aided by a graphical interface to increase understanding. To play the game as it is intended, they must also be able to think logically and make informed predictions. As it is a multiplayer game that connects to a server, computer literacy is also a useful skill. If the user wishes to connect a computer player to a game, they should also be familiar with the use of the command line.

## 2.4 Constraints

Full functionality of the application will be restrained by a required connection to a UofS server used for hosting games of Hanabi. User must have a valid NSID issued by the University of Saskatchewan to connect to the server.

## 2.5 Assumptions and Dependencies

The application will be implemented using the Java Development Kit and will be exported as a .Jar file. Any Windows, Mac, or Linux OS running the latest version of Java will execute as intended. Specific requirements can be found on the Java 8 system requirements page: https://www.java.com/en/download/help/sysreq.xml

# 3. Specific Requirements

## 3.1 External Interfaces

### 3.1.1 User Interfaces

Hanabi's Main Screen:

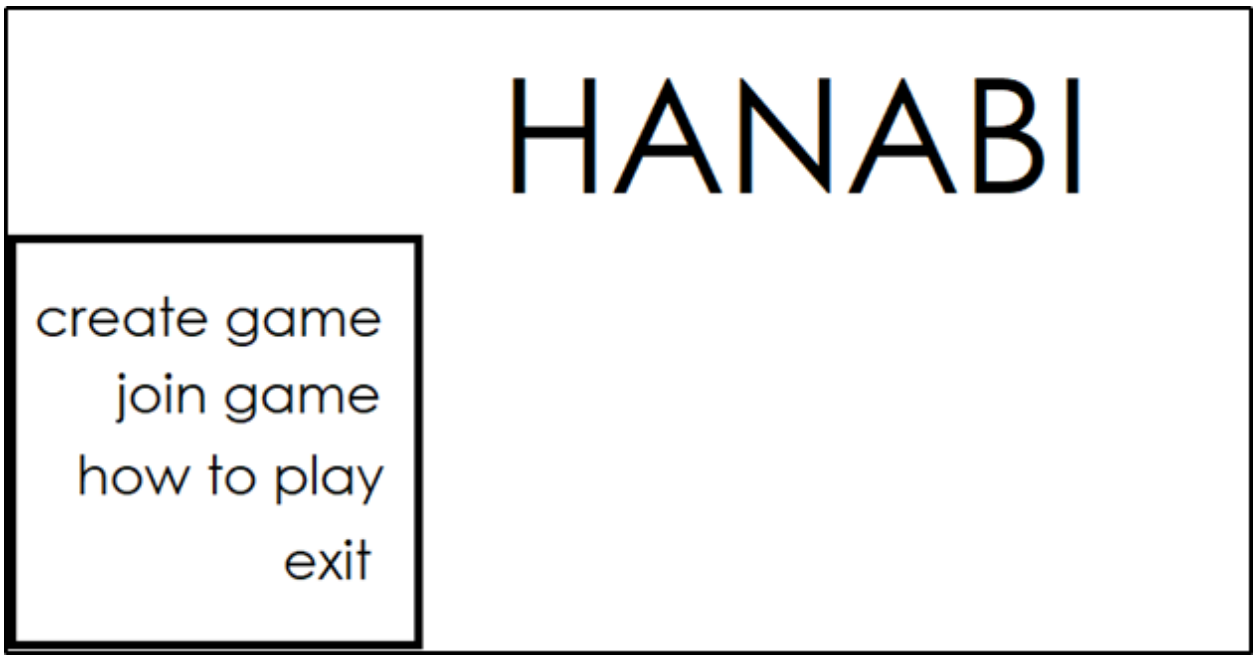


Figure 3

If user clicks “create game”:

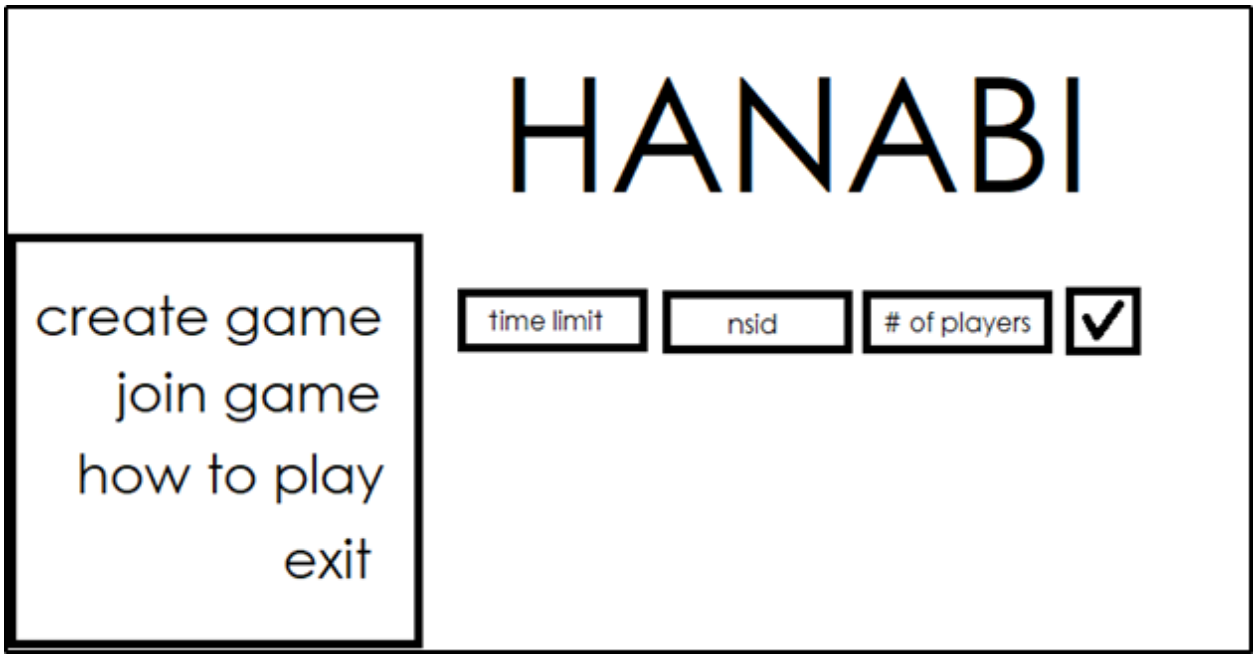


Figure 4

If user clicks “join game”:

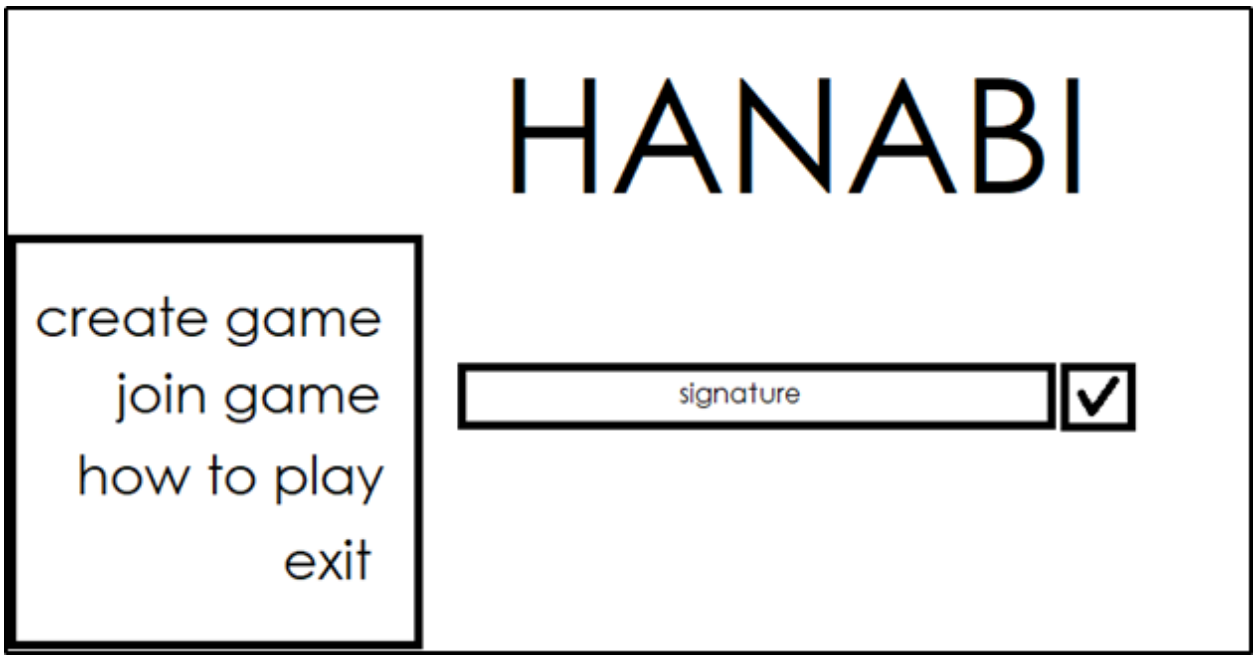


Figure 5

Waiting for other players to join game:

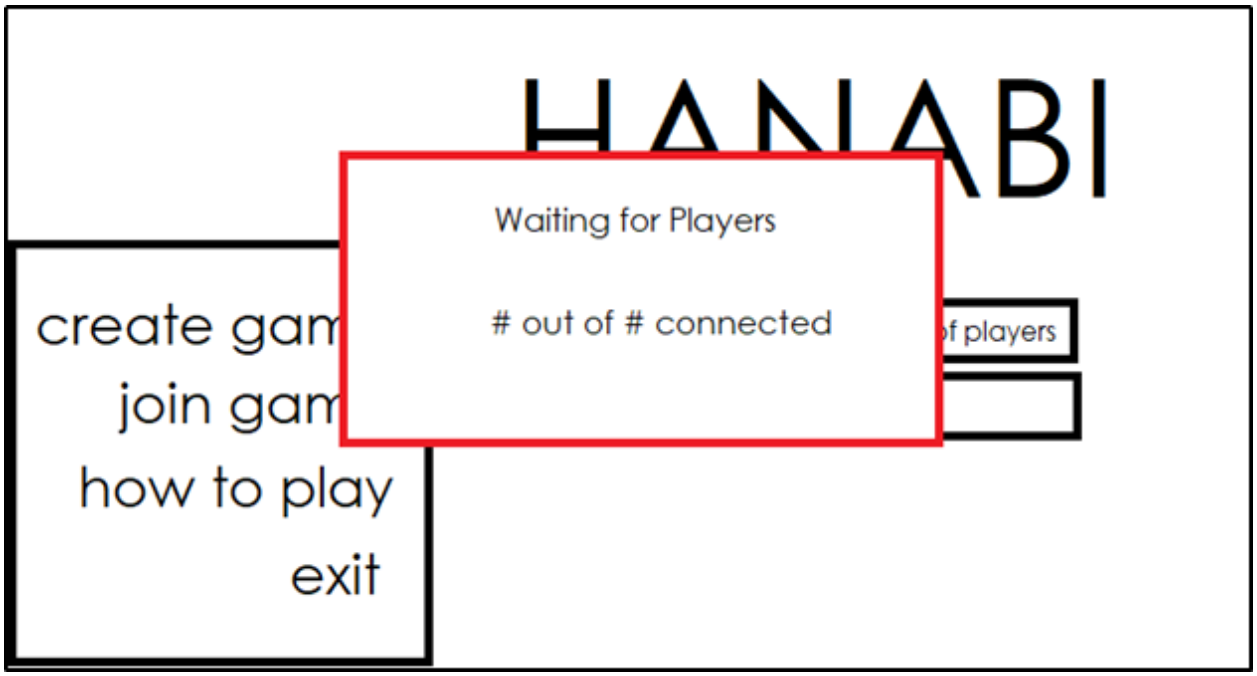


Figure 6

How to play pop-up:

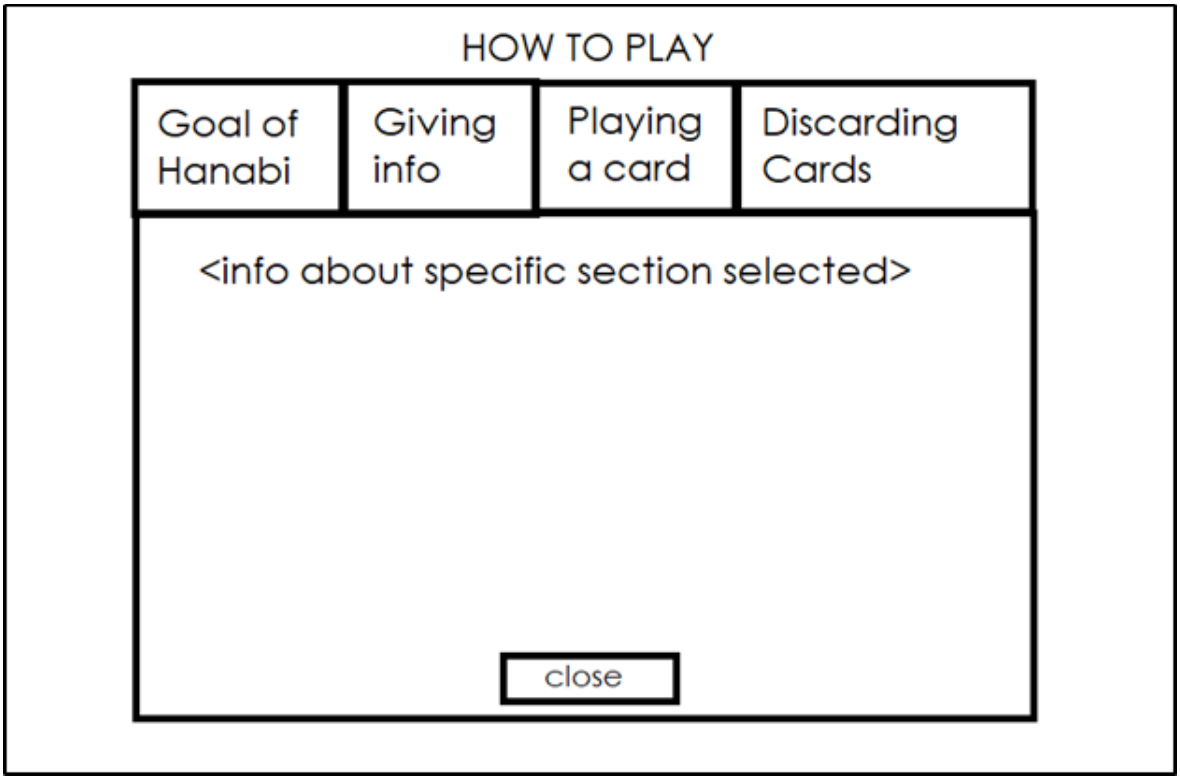


Figure 7

In-game screen:

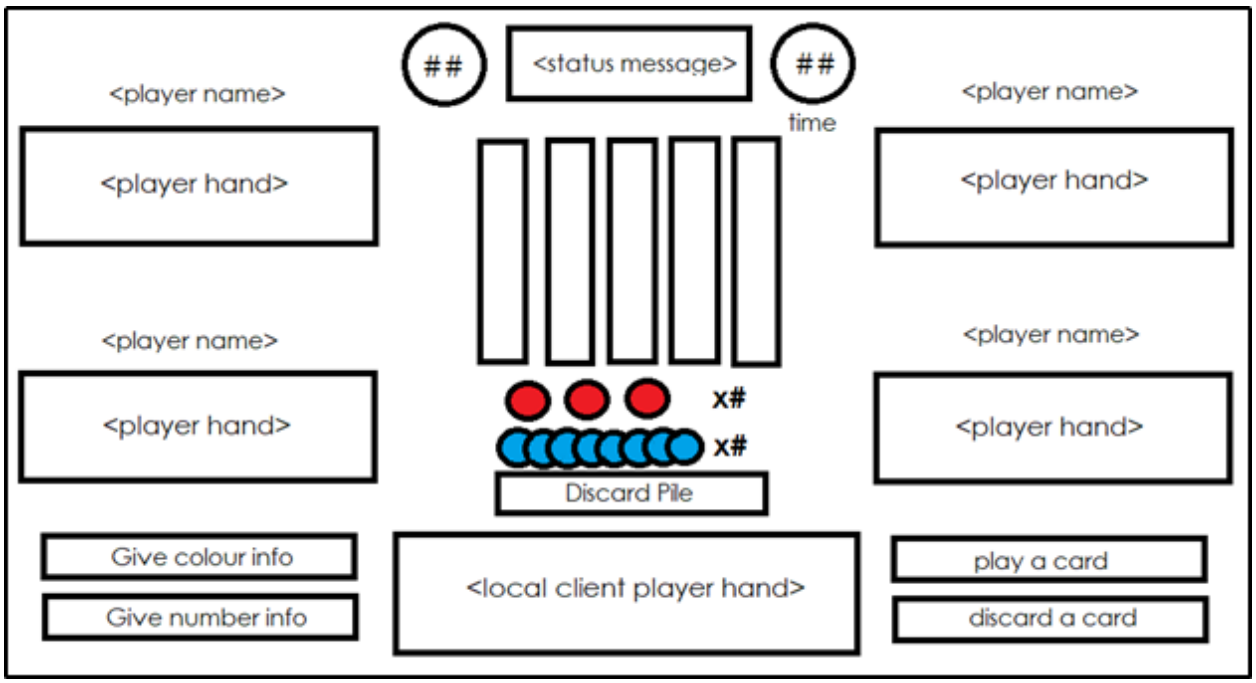


Figure 8

Mousing over cards during “Give colour info”:

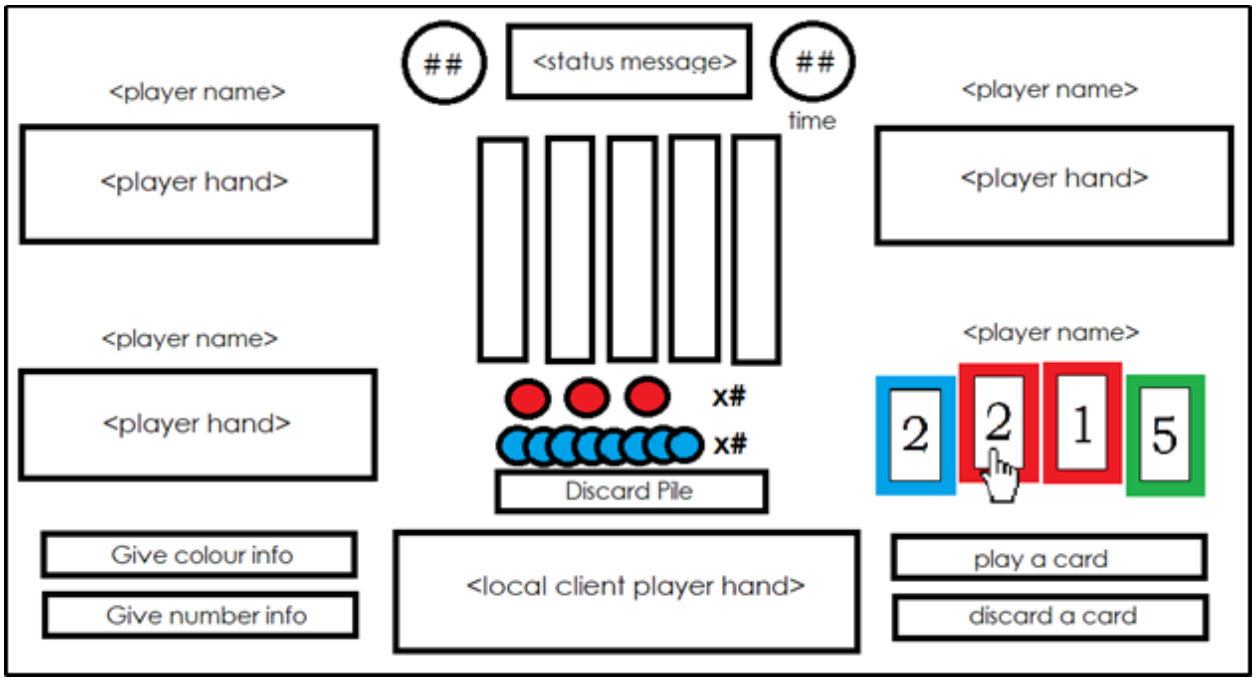


Figure 9

Mousing over cards during “Give number info”:

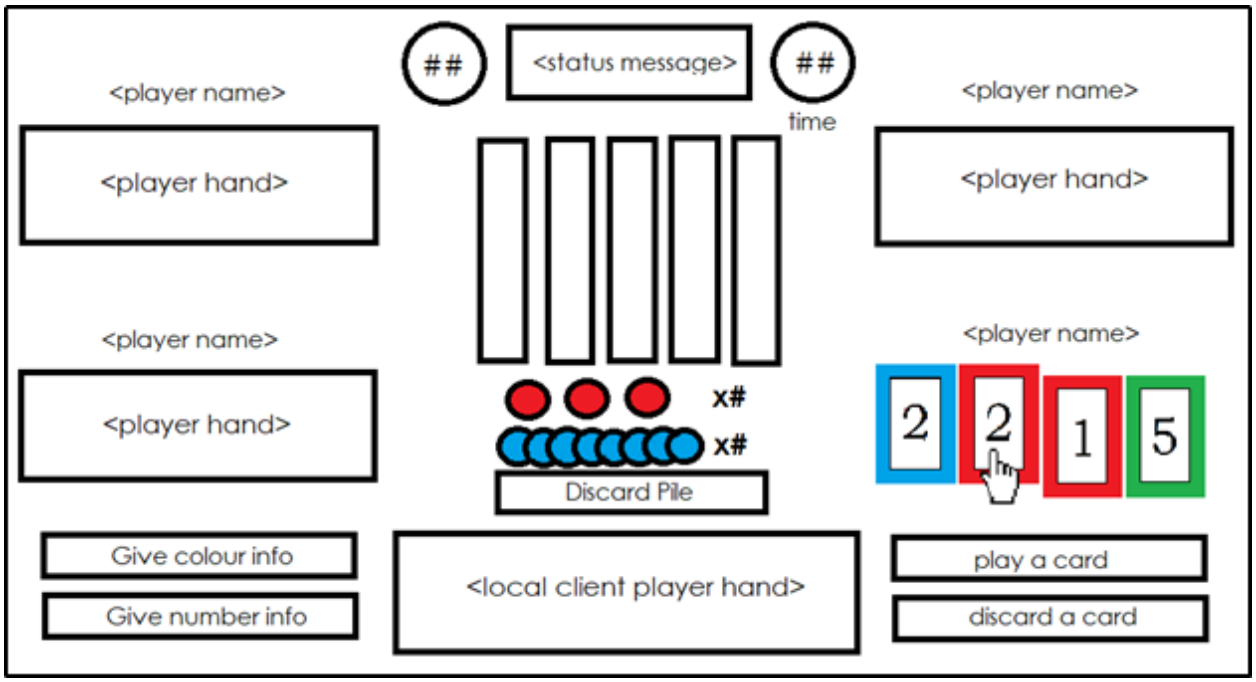


Figure 10

Discard Pile pop-up:

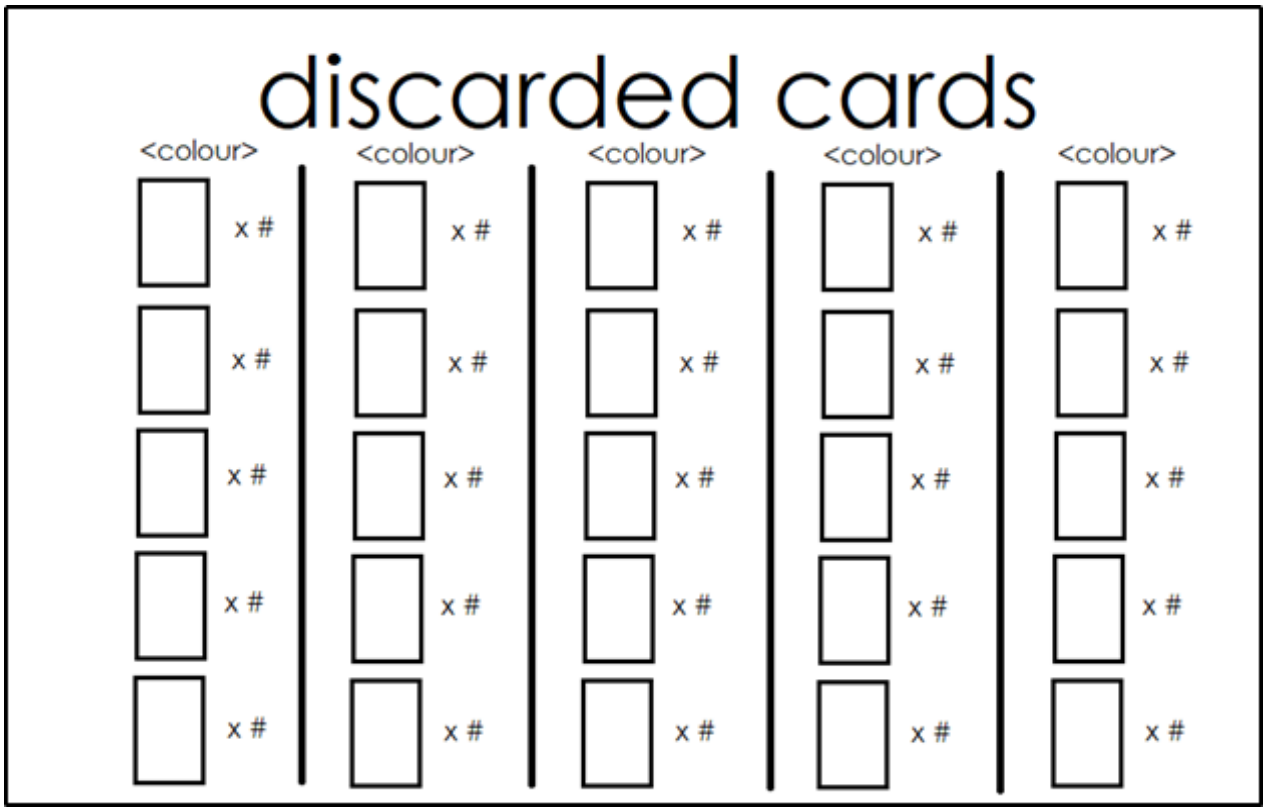


Figure 11

In-game screen with additional information:

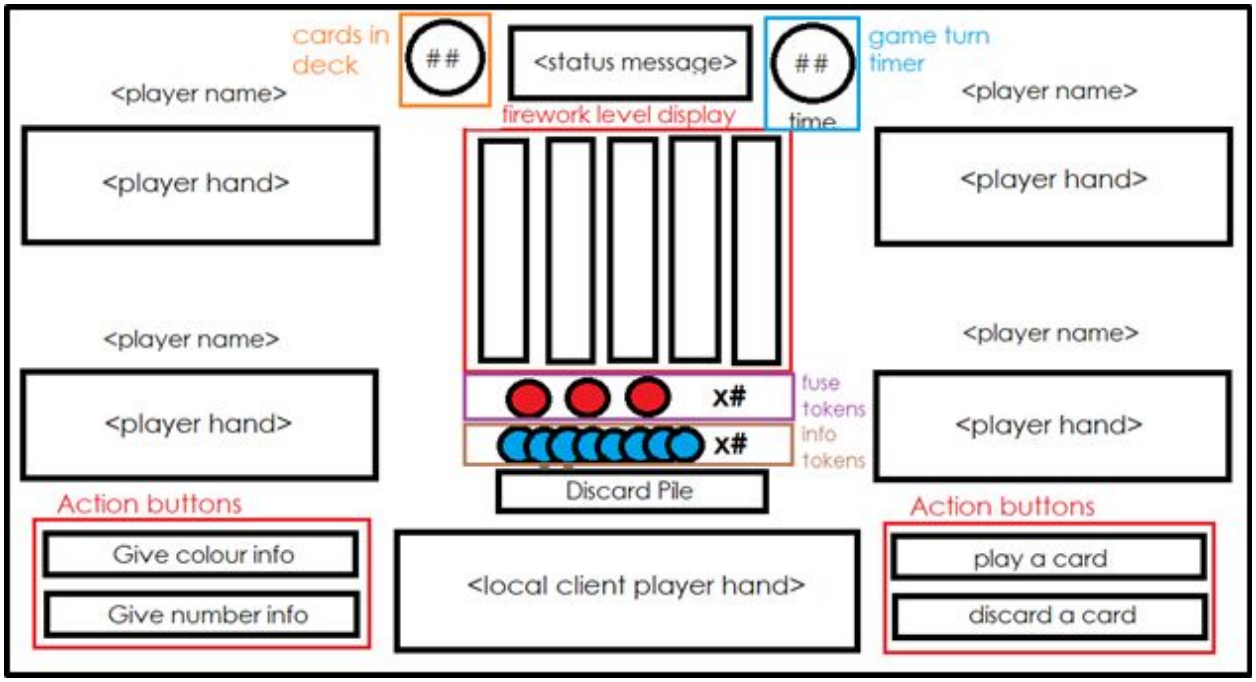


Figure 12

User's four card states: Unknown, Known Number, Known Colour, Known Card.

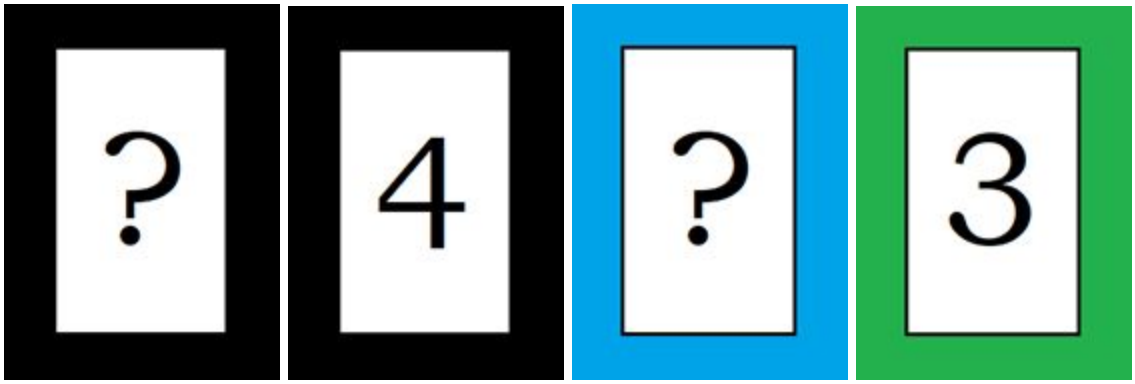


Figure 13:1-4

Information markers for other players' cards:

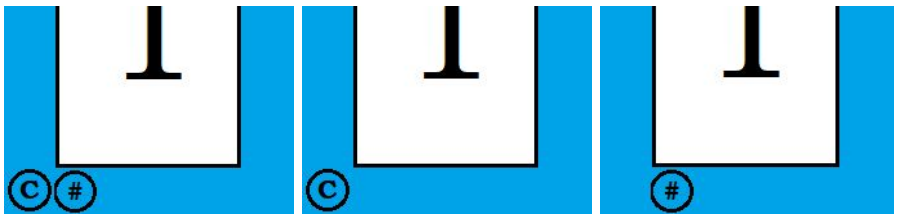


Figure 14: 1-3

Game ending screen:

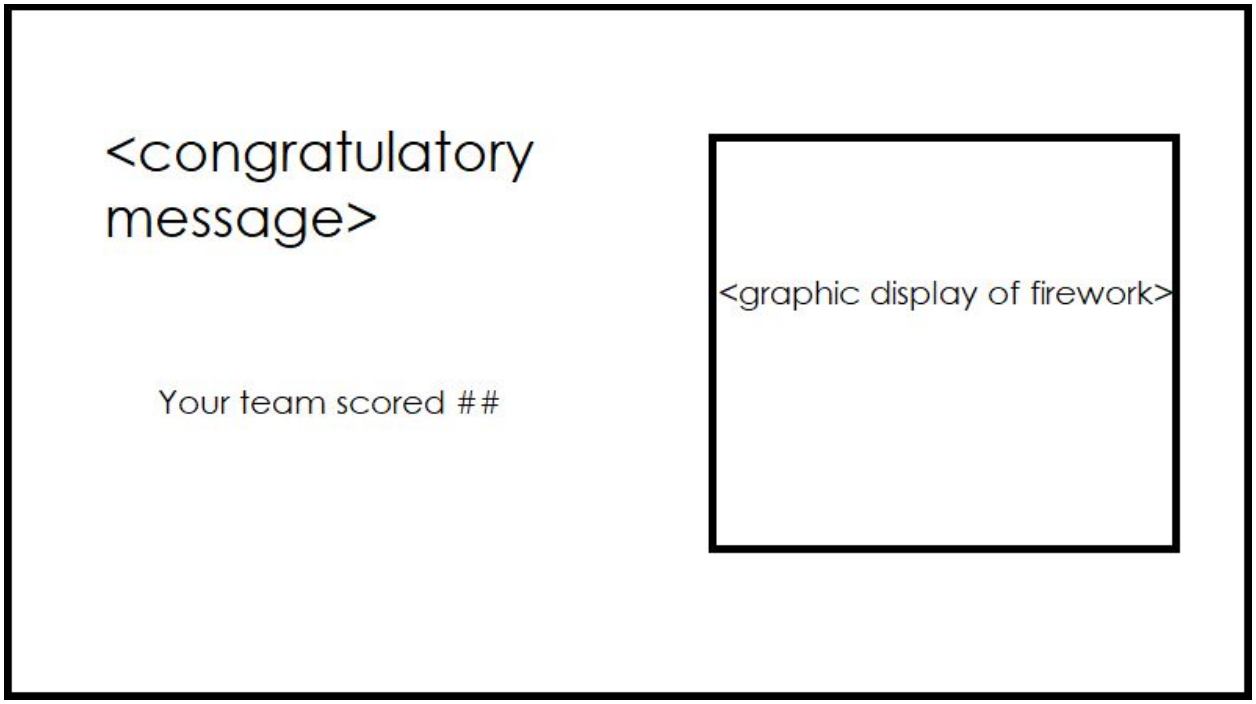


Figure 15

### 3.1.2 AI Interface

All commands and control of the AI will be executed on the command-line.

## 3.2 Functions

### 3.2.1 Use Cases

#### 3.2.1.1 Hosting a game

**Context of use:** Host player creates and joins a game lobby on the server which other players can join.
**Level:** User-goal
**Primary Actor:** Host player
**Precondition:** Must have an active connection to the server.
**Success End Condition:** Game has been started, host player joined the game and has the join key, server is ready for others players to join.
**Failed End Protection:** Game will not be created, host player notified and will remain at main menu.
**Trigger:** User decides to host a game.
**User Interface:** Reference figures 3, 4, and 6.
**Main Success Scenario**

1. Host player clicks “Host Game” button.
2. System displays text boxes for time out limit, NSID and # of Players.
3. Host player enters the information into the newly appeared text boxes.
4. Host player clicks “Ok” button.
5. System connects to the server.
6. System creates a game lobby on the server using the entered information.
7. Host player receives a signature key which other players will use to join the game.
8. Host player is moved to the in-game screen.

**Extensions**

4a. Host player provided invalid input in the text boxes: Display error message pop-up and prompt for server information.
5a. Host player cannot connect to the server: Display error message pop-up.

#### 3.2.1.2 Joining a game

**Context of use:** Joining player joins an existing game lobby.
**Level:** User-goal
**Primary Actor:** Joining player
**Preconditions:** Must have an active connection to the server, must have a valid join key.
**Success End Condition:** Joining player connects to the game.
**Failed End Protection:** Game will not be joined and will remain at main menu.
**Trigger:** User decides to join a game.
**User Interface:** Reference figures 3, 5, and 6.
**Main Success Scenario**

1. Joining player clicks “Join Game” button.
2. System displays text box for the join key.
3. Joining player enters the information into the newly appeared text box.
4. Joining player clicks “Join” button.
5. Joining player connects to the game lobby on the server using the entered information.
6. Joining player is moved to the in-game screen.

**Extensions**

3a. Joining player provided invalid input in the text box: Display error message pop-up and return joining player to the main menu.
5a. Joining player cannot connect to the server: Display error message pop-up and return joining player to the main menu.

#### 3.2.1.3 Reading the rules

**Context of use:** Player is shown the instructions and rules.
**Level:** User-goal
**Primary Actor:** Player
**Precondition:** None
**Success End Condition:** Player can read the instructions and rules.
**Failed End Protection:** None
**Trigger:** User decides to read the instructions and rules.
**User Interface:** Reference figures 3 and 7.
**Main Success Scenario**

1. Player clicks “How to Play” button.
2. System displays pop-up with clickable section headers.
3. Player clicks a specific section header.
4. Player is shown instructions pertaining to the clicked header.
5. Player clicks “Close” button.
6. System closes the pop-up instruction window.

**Extensions**

None

#### 3.2.1.4 Play a card

**Context of use:** Active Player plays a card in-game.
**Level:** User-goal
**Primary Actor:** Active player
**Preconditions:** Must be player's turn.
**Success End Condition:** Card from player's hand is added to a firework, a new card is dealt and turn is ended.
**Failed End Protection:** Card is discarded and a red launch counter is removed.
**Trigger:** User decides to play a card.
**User Interface:** Reference figure 8.
**Main Success Scenario**

1. Active player clicks "Play a card" button.
2. Active player clicks their card of choice.
3. Active player's card is removed from hand and placed onto the appropriate firework pile.
4. Active player is dealt another card face down.
5. Inactive players are notified of what card is played and what card is dealt.

**Extensions**

1a. User wants to cancel action: user can click same button to cancel choice, or another button to change to that choice
3a. Card cannot properly be played: Card is discarded and a red launch counter is removed.
4a. Deck is empty: Active player cannot draw new card.

#### 3.2.1.5 Discard a card

**Context of use:** Active Player discards a card in-game.
**Level:** User-goal
**Primary Actor:** Active player
**Preconditions:** Must be player's turn, all 8 info tokens cannot be in play.
**Success End Condition:** Card from player's hand is added to a discard pile, a new card is dealt, an info token is returned and turn is ended.
**Failed End Protection:** None
**Trigger:** User decides to discard a card.
**User Interface:** Reference figure 8.
**Main Success Scenario**

1. Active player clicks "Discard a card" button.
2. Active player clicks their card of choice.
3. Active player's card is removed from hand and placed onto the discard pile.
4. Active player is dealt another card face down.
5. Info token is returned to play.
6. Inactive players are notified of what card is discard and what card is dealt.

**Extensions**

1a. User wants to cancel action: user can click same button to cancel choice, or another button to change to that choice
6a. Deck is empty: Active player cannot draw new card.

#### 3.2.1.6 Give colour info

**Context of use:** Active Player gives colour info to an inactive player in-game.
**Level:** User-goal
**Primary Actor:** Active player
**Preconditions:** Must be player's turn, must be at least 1 info token in play.
**Success End Condition:** Colour info is given to specific player, info token is spent and turn is ended. Markers will be placed on informed cards for players other than the informed player.
**Failed End Protection:** None
**Trigger:** User decides to give colour info to another player.
**User Interface:** Reference figures 9 and 14.
**Main Success Scenario**

1. Active player clicks "Give colour info" button.
2. Active player hovers with the mouse over a card in any inactive players' hand.
3. Cards in the same player's hand which have the same colour as the one moused over are raised. Cards for which that colour information has already be shared do not raise.
4. Active player clicks moused over card.
5. Inactive player whose card is clicked is given colour info about all cards that were raised.
6. Inactive players whose cards weren't clicked are notified of what card info was given.
7. System removes one info token from play.

**Extensions**

1a. User wants to cancel action: user can click same button to cancel choice, or another button to change to that choice
3a. No cards raise: Colour info for moused over card or others that match it cannot be given.

#### 3.2.1.7 Give number info

**Context of use:** Active Player gives number info to an inactive player in-game.
**Level:** User-goal
**Primary Actor:** Active player
**Preconditions:** Must be player's turn, must be at least 1 info token in play.
**Success End Condition:** Number info is given to specific player, info token is spent and turn is ended. Markers will be placed on informed cards for players other than the informed player.
**Failed End Protection:** None
**Trigger:** User decides to give number info to another player.
**User Interface:** Reference figures 10 and 14.
**Main Success Scenario**

1. Active player clicks "Give number info" button.
2. Active player hovers with the mouse over a card in any inactive players' hand.
3. Cards in the same player's hand which have the same number as the one moused over are raised. Cards for which that number information has already be shared do not raise.
4. Active player clicks moused over card.
5. Inactive player whose card is clicked is given number info about all cards that were raised.
6. Inactive players whose cards weren't clicked are notified of what card info was given.
7. System removes one info token from play.

**Extensions**

1a. User wants to cancel action: user can click same button to cancel choice, or another button to change to that choice
3a. No cards raise: Number info for moused over card or others that match it cannot be given.

#### 3.2.1.8 Look at discard pile.

**Context of use:** Player is shown all the cards in the discard pile.
**Level:** User-goal
**Primary Actor:** Player (Both active and inactive)
**Precondition:** None
**Success End Condition:** Player can look at discard pile.
**Failed End Protection:** None
**Trigger:** User decides to look at discard pile.
**User Interface:** Reference figure 11.
**Main Success Scenario**

1. Player clicks "Discard Pile" button.
2. System displays pop-up containing all cards in the discard pile.
3. Player clicks "Close" button.
4. System closes the pop-up instruction window.

**Extensions**

None

#### 3.2.1.9 Player disconnects

**Context of use:** Player exits program while in a game.
**Level:** User-goal
**Primary Actor:** Player (Both active and inactive)
**Precondition:** None
**Success End Condition:** Player's program is closed, other players are returned to main menu.
**Failed End Protection:** None
**Trigger:** User decides to exit game.
**User Interface:** NA
**Main Success Scenario**

1. Player clicks "Close".
2. System closes the players program
3. Other players are moved back to main menu, game is ended, server is closed and other players are shown the score screen.
4. Other players click "Close" on score screen.

**Extensions**

None

#### 3.2.1.10 Player exits program

**Context of use:** Player exits program while not in a game.
**Level:** User-goal
**Primary Actor:** Player
**Precondition:** None
**Success End Condition:** Player's program is closed.
**Failed End Protection:** None
**Trigger:** User decides to exit program.
**User Interface:** NA
**Main Success Scenario**

1. Player clicks "Close".
2. System closes the players program.

**Extensions**

None

## 3.3 Performance Requirements

The system shall support a minimum of two players (AI or Human) and a maximum of five.
The system shall support hosting one game per valid NSID, but allow multiple simultaneous game connections per NSID.
The AI shall be built to complete a turn within one second.

## 3.4 Design Constraints

There are no specific design constraints on this program.

## 3.5 Software System Attributes

### 3.5.1 Reliability

The system shall properly respond to a human player's inputs at all times, and the game state will always remain correct and up to date on a human player's client. In the case of unusual or unexpected errors the system shall be robust enough to handle them or fail gracefully.

### 3.5.2 Availability

The availability of the client running on a player's machine shall always be 100% assuming the client system has not undergone significant change since successfully running it initially. The server is an external factor that we can expect but not guarantee 100% uptime and availability on. This also assumes the availability of an internet connection on the client computer.

### 3.5.3 Security

The system shall use student NSID's from the University of Saskatchewan, unique tokens, signatures, game ids, and passwords to ensure connection security. With those security measures in place the host can be confident in the fact that only those they invite can connect to the game.

### 3.5.4 Maintainability

The application will be built in a way that allows for minor extensions and future adjustments. Code will be well documented and coupling will be limited as much as possible to allow for easy reuse and refactoring.

### 3.5.5 Portability

The application will be runnable on many systems with the ability to install Java Runtime Environment and by extension to run .jar files. The baseline will be the hardware in the Spinks computer laboratory machines and further system compatibility will be desired but not necessary.

# 4. Supporting Information

## 4.1 Appendices

Appendix A - Omitted Sections

The following sections have been omitted from the document:

- Software Interfaces - There are no specific interfaces with other software.
- Hardware Interfaces - Minimal interfaces with hardware.
- Memory constraints - No memory constraints of note.
- Site Adaptation Requirements - Future versions are not planned.
- Logical Database Requirements - External databases are unsupported.
- Design Constraints - Design constraints are insignificant and not need.