Title page

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 to along with human players at the same time. The game will be playable online through a dedicated server.

1.3 Definitions, Acronyms, and Abbreviations

"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.

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 at the following link for those in need of a refresher:

http://www.boardgamecapital.com/game\_rules/hanabi.pdf

1.5 Overview //of document

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 ran on a user’s local machine to host or join games of Hanabi on 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

The two part 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.

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.

2.1.3 Hardware Interfaces

The software does not require additional hardware interfaces to function.

2.1.4 Software Interfaces

The software does not require additional software interfaces to function.

2.1.5 Communications Interfaces

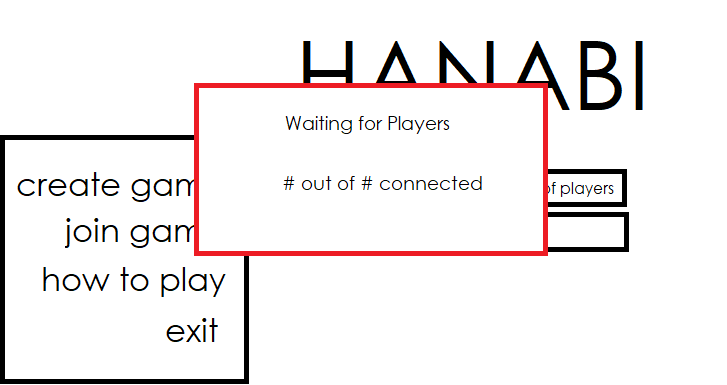
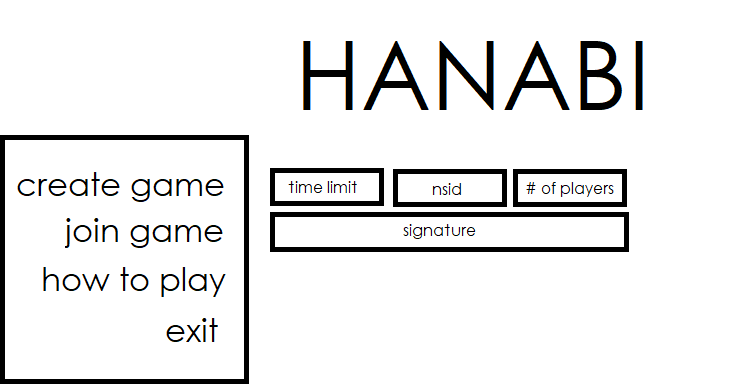
The software requires TCP in order to communicate with the Hanabi server.

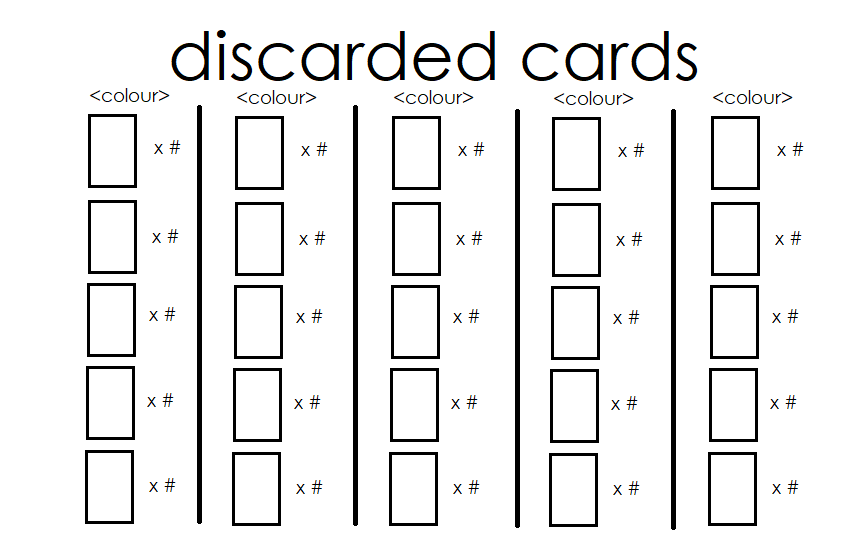
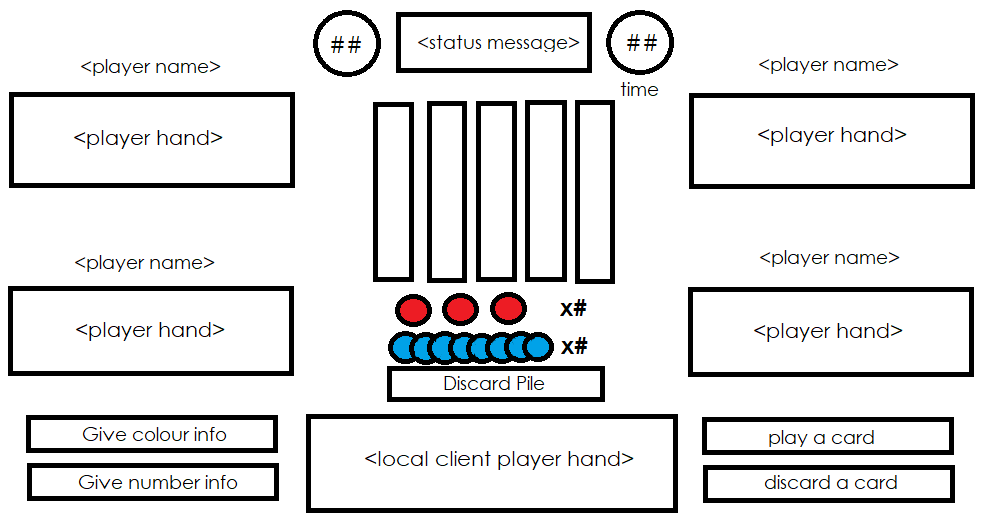
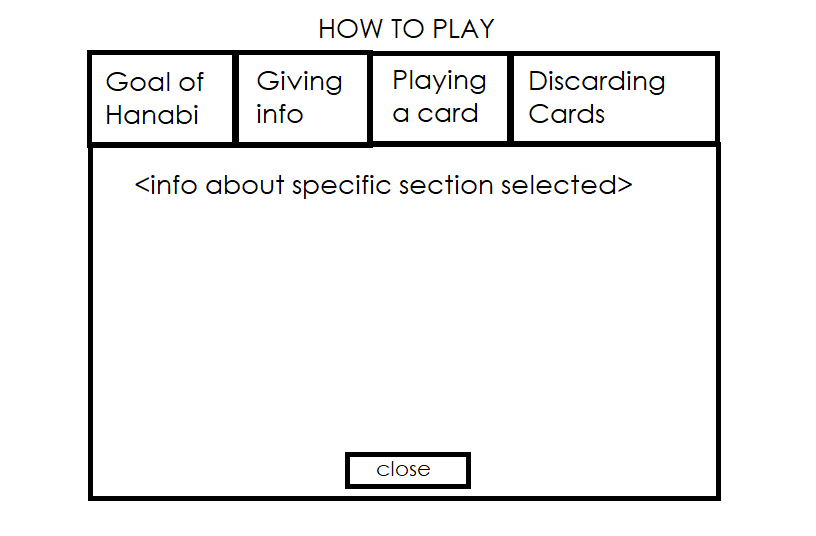
2.1.6 Memory Constraints

The software does not have any specific memory constraints.

2.1.7 Operations

2.1.8 Site Adaptation Requirements





Hardware Interfaces

Software Interfaces - "Omit?" #No interfaces w/ other SW

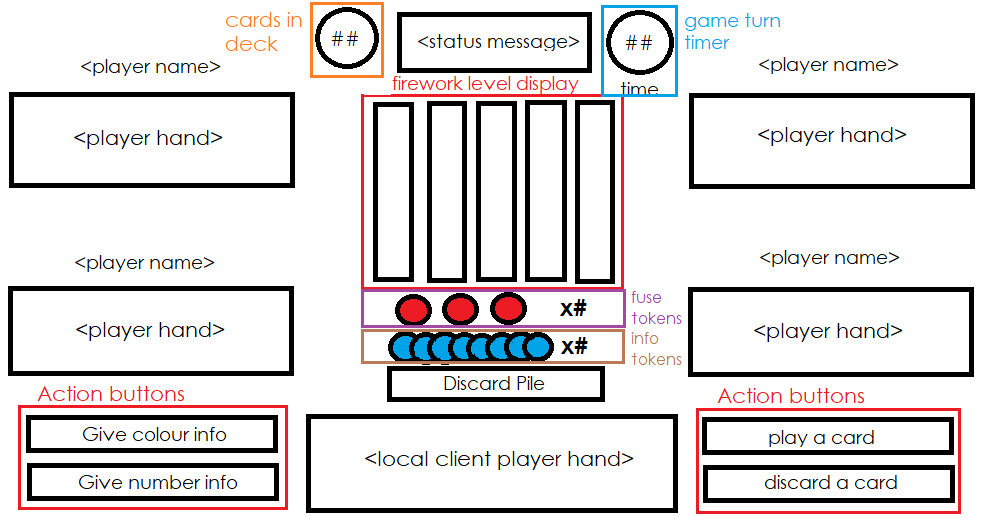
Communications Interfaces

Memory Constraints - "Omit?" #No specific memory constraints

Operations - "Omit?" #user interfaces will cover

Site Adaptation Requirements - "Omit?" #No site uses

2.2 Product Functions - IP Troy



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.

3 Specific Requirements

3.1 External Interfaces

3.2 Functions

3.3 Performance Requirements

3.4 Logical Database Requirements

3.5 Design Constraints

Standard Compliances - "Omit?"

3.6 Software System Attributes

3.6.1 Reliability

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

3.6.2 Availability

The availability of the client running on a players 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.

3.6.3 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.

3.6.4 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.

3.6.5 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.

3.7 Organizing the Specific Requirements

System Mode

User Class

Objects

Stimulus

Response

Functional Hierarchy

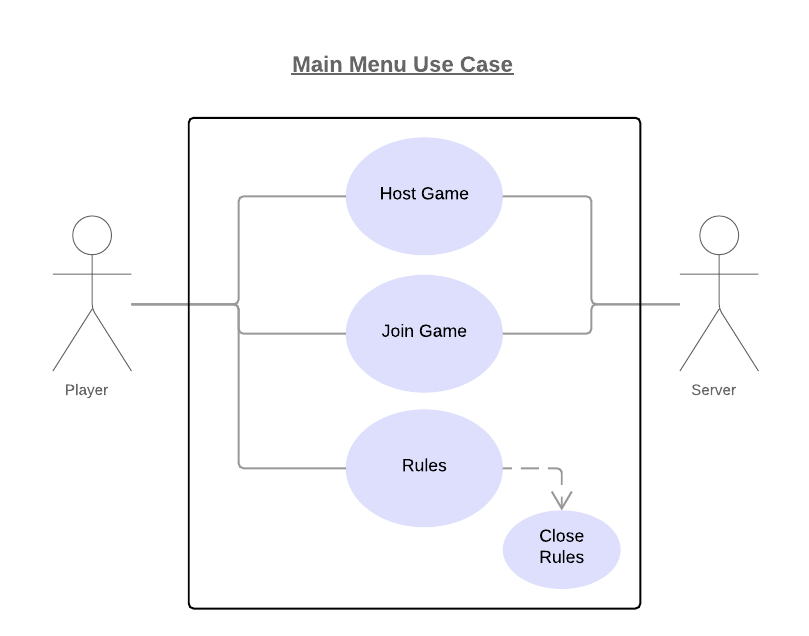
Additional Comments

4 Supporting Information

4.1 Table of Contents and Index

4.2 Appendices

#SRS sections are detailed in the ISO 830-1998 pgs.11-20



**Use Case 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.

**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 “Create” 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.

**Use Case 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.

**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.

**Use Case 3:** Reading How to Play.

**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.

**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.

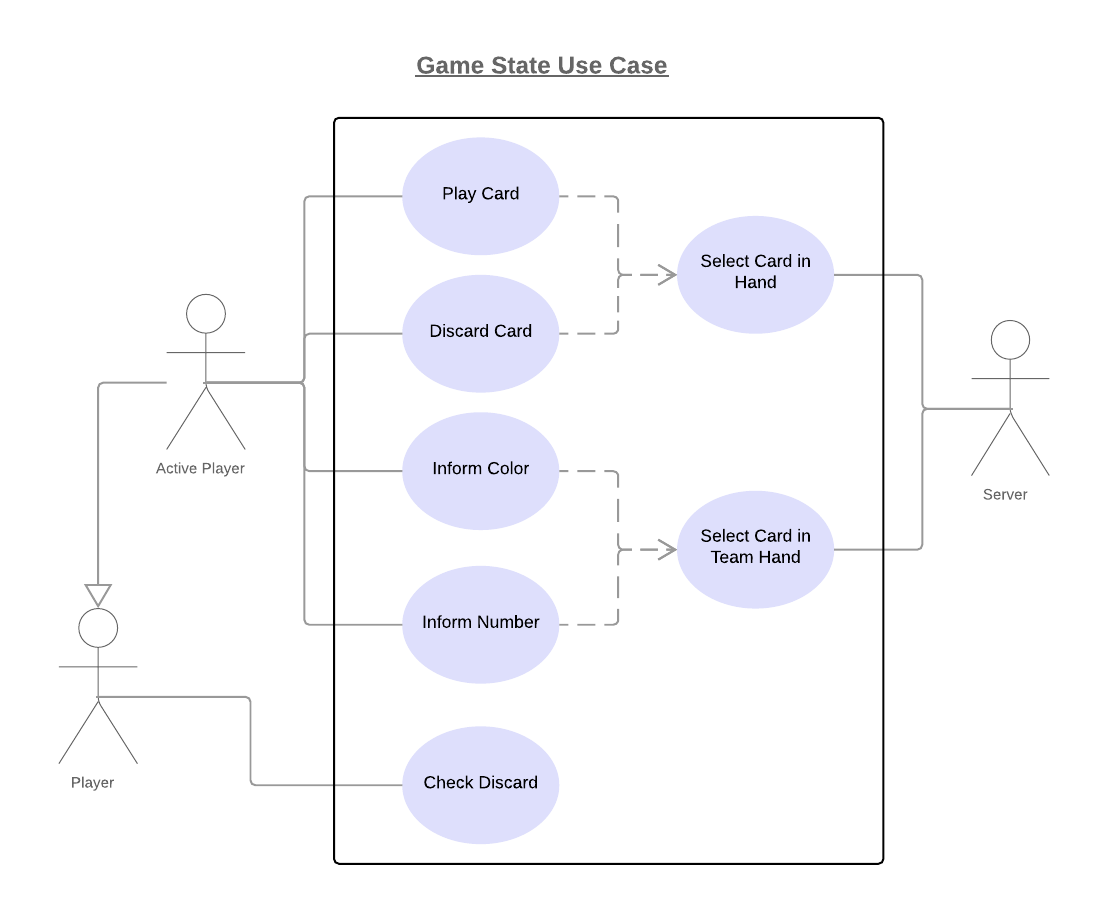
4. Player is shown instructions pertaining to the clicked header.

5. Player clicks “Close” button.

6. System closes the pop-up instruction window.

**Extensions**

None



**Use Case 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.

**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**

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.

**Use Case 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.

**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**

6a. Deck is empty: Active player cannot draw new card.

**Use Case 6:** Give colour info.

**Context of use:** Active Player give 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.

**Failed End Protection:** None

**Trigger:** User decides to give colour info to another player.

**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**

3a. No cards raise: Colour info for moused over card or others that match it cannot be given.

**Use Case 7:** Give number info.

**Context of use:** Active Player give 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.

**Failed End Protection:** None

**Trigger:** User decides to give number info to another player.

**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**

3a. No cards raise: Number info for moused over card or others that match it cannot be given.

**Use Case 8:** Looking 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.

**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