Rummy

Software Requirement Specification

**1. Introduction**

**1.1. Purpose**

Virtual rummy game for anyone interested in card games.

**1.2. Scope**

This will be a C++ environment for a 2 player game of rummy, the real player and an AI.

**1.3. Definitions, acronyms and abbreviations**

AI- Artificial Intelligence player

GUI- Graphic User Interface

Set- Group of cards such that the cards make up a group of 4 numbers (ie 7 of clubs, hearts, diamonds, and spades)

Sequence- a group of cards such that a suit is the same and goes in numerical order, (ie one, two, three, four, all of hearts)

**1.4. References**

<https://www.pagat.com/rummy/rummy.html#basic>

**1.5. Overview**

The remainder of the document will include a chapter 2 and chapter 3, Overall Description and Specific Requirements. Overall Description will describe some of the informal requirements so that the technical requirements described in the Specific Requirements section is easier to understand.

**2. Overall Description**

**2.1. Product perspective**

This will be one game of four included in an environment where each game can be played, however each game does not interact with one another.

**2.2. Product function**

A user will have a set turn where they must draw, then have the choice between placing a run or a book, adding to an existing meld, or discarding a card

**2.3. User characteristics**

User knows how to play (gin) rummy.

User knows the terminology of different aspects of rummy (meld, stock, discard....)

**2.4. Constraints**

No betting real money.

**2.5. Assumptions and dependencies**

The environment requires bulletproofing in the case that the player does not know how to play rummy and will need to prevent the player from making illegal moves.

**2.6. Requirements subsets**

UI pictures such as faces of cards can be dropped if time runs short, cards do not need to look pretty in order to play the game.

**3. Specific Requirements**

**3.1. External interface requirements**

**3.1.1. User interfaces**

The user will be given buttons and cards that will act as buttons where a user can click on them using a mouse. The game will be done by menu-driven actions.

**3.1.2. Hardware interfaces**

The software needs to output a ui on a monitor, moves cannot simply be done in a background process, and a user must be able to interact with the gui using a mouse.

**3.1.3. Software interfaces**

The software will be communicating with the operating system as well as something to compile the c++ program.

**3.1.4. Communications interfaces**

No other communication interfaces will be used, other players will be ai programs

**3.2. Functional requirements**

**3.2.1. Turn Start**

Introduction: The player starts their turn.

Inputs: A click on the card you wish to draw (from deck or from discard pile).

Processing: This is the draw action so the player clicks on the card they want to draw and the card will then move to the hand that only they can see.

Outputs: The selected card is moved and in the player’s hand.

**3.2.2. Meld**

Introduction: After the player draws a card, they choose to meld.

Inputs: A mouse click on the selected cards in hand the player wishes to meld.

Processing: The player clicks on all cards they wish to meld. Then click on an open spot in the meld area of the GUI and the cards move there if it’s a valid move.

Outputs: The cards are now in the selected spot in the meld area.

**3.2.3 Lay Off**

Introduction: After a card is drawn or after the first meld/lay off or after subsequent lay off, the player decides to lay off.

Inputs: One card is selected from the player’s hand.

Processing: The one card selected, then the player clicks on the meld they wish to add the card to and the card moves there.

Outputs: the selected card is in the selected meld.

**3.2.4 Discard**

Introduction: After the player draws a card or after the first meld/lay off or after subsequent lay off, they decide to discard.

Inputs: The player selects the card in their hand they wish to discard.

Processing: A card is selected from the player’s hand. They then click the discard pile on the GUI and the card moves out of the hand to the pile.

Outputs: The card is in the discard pile and out of the player’s hand.

**3.2.5 Opponent’s Turn**

Introduction: It becomes the opponent’s turn when the current player discards a card and still has one or more cards left in their hand.

Inputs: A card is placed on the discard pile.

Processing: After the card is placed in the discard, the program will check to make sure the current number of cards in the player’s hand is one or more.

Outputs: Since there are one or more cards in the user’s hand, it becomes the opponent's turn.

**3.2.6 Game Over**

Introduction: The game is over when the player discards a card and has no more cards left.

Inputs: A card is placed on the discard pile.

Processing: After the card is placed in the discard, the program will check to make sure there are no more cards in the player’s hand.

Outputs: The program says the game is over and who the winner is.

**3.2.7 Legal Move Check**

Introduction: When each player makes a move, the program will check if it is a legal move.

Inputs: The player’s choosen move.

Processing: The program will take the player’s selected move and make sure it was legal or illegal.

Outputs: The program will either let the player continue in the game if the move was legal or ask the player to redo the move if it was illegal.

**3.2.8 Scoring**

Introduction: In order for a winner to be confirmed, the program will have to keep track of both player’s scores.

Inputs: Each player’s scores based on what they did each turn.

Processing: The program will keep track of what each player does durning their turn and add up scores accordingly.

**3.3. Performance requirements**

Performance requirements will be minimal. Your machine must be able to run c++ programs.

**3.4. Design Constraints**

The computer must have memory space required for the program on a hard drive.

**3.4.1. Standards compliance**

Must not break gambling laws

**3.4.2. Hardware limitations**

Hardware will not be an issue in most to all cases

**3.5. Software system attribute**

**3.5.1. Availability**

The software likely won't have issues with the operating system that would cause faults

**3.5.2. Security**

The product will not attempt to break into the computer’s sensitive functions.

**3.5.3. Maintainability**

The product could be ported to mac os and linux for future compatibility, graphics could be upgraded to look better in the future, animations associated with things such as drawing a card, placing a meld, ext

Gin Rummy

Software Requirement Specification

**1. Introduction**

**1.1. Purpose**

Creating an environment that allows a user to play gin rummy against an ai opponent. This product will be for those interested in card games, or for those who want to practice gin to get better.

**1.2. Scope**

The product will be a virtual environment to play gin rummy.

**1.3. Definitions, acronyms and abbreviations**

AI- Artificial Intelligence player

GUI- Graphic User Interface

Set- Group of cards such that the cards make up a group of 4 numbers (ie 7 of clubs, hearts, diamonds, and spades)

Sequence- a group of cards such that a suit is the same and goes in numerical order, (ie one, two, three, four, all of hearts)

Deadwood- Cards leftover after melding

**1.4. References**

<https://www.pagat.com/rummy/ginrummy.html>

**1.5. Overview**

This document will explain the product design, requirements, and process of a user in the software as part of a game of gin rummy.

**2. Overall description**

**2.1. Product perspective**

The product will be created alongside 3 other games, and a menu will be used to navigate between each.

**2.2. Product function**

A player will be prompted to draw, then given the option to knock or discard, then the ai player goes. This sequence continues until a player knocks.

**2.3. User characteristics**

The user should know the rules of gin rummy.

The user should be familiar with terminology of different aspects of gin rummy (draw,discard,knocking,sets and runs,going gin....)

**2.4. Constraints**

No betting money.

**2.5. Assumptions and dependencies**

The environment requires bulletproofing in the case that the player does not know how to play rummy and will need to prevent the player from making illegal moves.

**2.6. Requirements subsets**

Ui pictures such as faces of cards can be dropped if time runs short, cards do not need to look pretty in order to play the game.

**3. Specific Requirements**

**3.1. External interface requirements**

**3.1.1. User interfaces**

The user will be given buttons and cards that will act as buttons where a user can click on them using a mouse. The game will be driven by menu-driven actions.

**3.1.2. Hardware interfaces**

The software needs to output a ui on a monitor, moves cannot simply be done in a background process, and a user must be able to interact with the gui using a mouse.

**3.1.3. Software interfaces**

The software will communicate with the operating system as well as something to compile the c++

**3.1.4. Communications interfaces**

No other communication interfaces will be used, other players will be ai programs

**3.2. Functional requirements**

**3.2.1. Turn Start**

Introduction: The player starts their turn.

Inputs: A click on the card you wish to draw.

Processing: This is the draw action so the player clicks on the card they want to draw and the card will then move to the hand that only they can see.

Outputs: The selected card is moved and in the player’s hand.

**3.2.2. Discard**

Introduction: After the player has selected the card, a card must be discarded to complete a turn ( A player cannot discard the last card drawn).

Inputs: The player selects any other card in their hand they wish to discard.

Processing: A card is selected from the player’s hand. They then click the discard pile on the GUI and the card moves out of the hand to the pile.

Outputs: The card is in the discard pile and out of the player’s hand.

**3.2.2. Knocking**

Introduction: After the player draws a card, a player can end the game by knocking

Inputs: The player clicks on the “knock” button in the GUI.

Processing: When the “knock” button is pressed, the game goes to the end game sequence. In order to Knock, that players hand must be able to be split into runs or sets. All unmatched cards must be below 10.

Outputs: The end game sequence will be followed.

**3.2.3 Opponent’s Turn**

Introduction: It becomes the opponent’s turn when the current player discards a card and still has one or more cards left in their hand in the game sequence and end game sequence.

Inputs: A card is placed on the discard pile.

Processing: After the card is placed in the discard, the program will check to make sure the current number of cards in the player’s hand is one or more.

Outputs: Since there are one or more cards in the user’s hand, it becomes the opponent's turn.

**3.2.4 End Game- Meld**

Introduction: Durning the end game, a player can choose to meld.

Inputs: The player will select the cards they wish to meld and an empty spot in the meld are of the GUI.

Processing: When the player selects the card and then the meld area, the program will only move the cards to that selected area.

Outputs: The selected cards have successfully moved to the chosen spot in the meld area of the GUI.

**3.2.5 End Game- Discard**

Introduction: A player chooses to discard after a player has knocked or melded

Inputs: a user clicks on a card, then the discard pile

Processing: The program must ensure that the discard is a legal move, a card cannot be discarded if a player “goes gin”

Outputs: The card appears in the discard pile.

**3.2.6 End Game Opponent w/o deadwood**

Introduction: After the player discards a card and has no deadwood, the opponent has some options.

Inputs: The opponent can choose to select cards to meld and or discard some cards.

Processing: The game will see if the opponent can meld cards and/or just discard one.

Outputs: No matter what the opponent will have less cards in their hand.

**3.2.7 End Game Opponent w/ deadwood**

Introduction: After the player discards a card and has some deadwood, the opponent has some options.

Inputs: The opponent can choose to select cards to meld, lay off onto the player’s melds or discard.

Processing: The program will see if the opponent can meld, lay off, or discard.

Outputs: The opponent will have less cards in their hand.

**3.2.8 Legal Move Check**

Introduction: When each player makes a move, the program will check if it is a legal move.

Inputs: The player’s choosen move.

Processing: The program will take the player’s selected move and make sure it was legal or illegal.

Outputs: The program will either let the player continue in the game if the move was legal or ask the player to redo the move if it was illegal.

**3.2.9 Scoring**

Introduction: In order for a winner to be confirmed, the program will have to keep track of both player’s scores.

Inputs: Each player’s scores based on what they did each turn.

Processing: The program will keep track of what each player does during their turn and add up scores accordingly.

Outputs: The appropriate scores will be shown.

**3.3. Performance requirements**

Performance requirements will be minimal. Your machine must be able to run c++ programs.

**3.4. Design Constraints**

The computer must have memory space required for the program on a hard drive.

**3.4.1. Standards compliance**

The program must not break gambling laws

**3.4.2. Hardware limitations**

Hardware will not be an issue in most to all cases

**3.5. Software system attribute**

**3.5.1. Availability**

The software likely won't have issues with the operating system that would cause faults

**3.5.2. Security**

The software will not attempt to access sensitive files. Otherwise not much security is needed for the program itself.

**3.5.3. Maintainability**

The product could be ported to mac os and linux for future compatibility, graphics could be upgraded to look better in the future, animations associated with things such as drawing a card, placing a meld, ext