Poker++

PROJECT GROUP 002-5

3/18/2020

# Introduction

## Group Members

|  |  |
| --- | --- |
|  | Members of the project group are Shishir Acharya, Randall Ferree and Robbie Frazier. Member contributions are summarized below:   * S.Acharya: System models (state diagram) * R.Ferree: GUI and events list * R.Frazier: Requirements table and system models (context and activity diagrams) |

## Project Overview

|  |  |
| --- | --- |
|  | The scope of this project is to design and develop a multiplayer 5 Card Draw Poker software system. The system will consist of separate applications for a dealer and multiple players and will support inter-application communication. The rules of the game are based on the information found in the video at the address below.  <https://www.youtube.com/watch?v=UmtSUhSfyYE>  The primary requirements were derived from a project description document provided by the primary stakeholder – Mr. Bud Davis, instructor of the CSE-3310-002 course during the Spring 2020 semester at the University of Texas at Arlington. Additional requirements were determined by in-class discussion and collaboration among the members of this group. |

# Requirements

## Requirements Table



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | See Requirements Table file   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **DESCRIPTION** | **TEST CASE** | **SOURCE** | **CLASS / METHOD** | **NOTES** | | The system shall accommodate exactly one dealer process | A second instance of the dealer application will fail to open on the same port | Project Description | Dealer::Dealer() Game::dealer |  | | The dealer shall control the deck of cards |  | Analysis | Game::deck |  | | The system shall accommodate two, three, four or five player processes | Gameplay will not start with fewer than two player processes The dealer will not accept a sixth player process until an existing player has left the game | Project Description | Game::player\_join() Game::players Player::Player() |  | | Each player shall have a unique identifier |  | Project Description | Player::UUID |  | | Players shall only be able to see the face values of cards in their own hand |  | Video | Player (GUI) |  | | The card deck shall consist of 52 cards |  | Video | Game::deck |  | | The card deck shall consist of 4 suits: hearts, spades, diamonds, clubs |  | Video | Card::Suit |  | | Each suit shall consist of cards Ace, 2, 3, 4, 5, 6, 7 ,8 ,9, 10, Jack, Queen, King |  | Video | Card::Card\_value |  | | Ace can be high (above King) or low (below 2) |  | Video | Dealer::determine\_winner() |  | | The deck shall be shuffled by the dealer three times at the beginning of each hand |  | Video | Dealer::shuffle() |  | | Cards shall be dealt by the dealer from the deck one at a time to each player in sequential order clockwise from left until each player has five cards |  | Video | Dealer::deal() |  | | The system shall accommodate a discard pile for all unwanted or folded cards |  | Analysis | Dealer::exchange() |  | | ~~There shall be three denominations of betting chips~~ |  | ~~Video~~ |  | ~~Obsoleted~~ | | ~~White chips shall have a value of $1~~ |  | ~~Video~~ |  | ~~Obsoleted~~ | | ~~Red chips shall have a value of $5~~ |  | ~~Video~~ |  | ~~Obsoleted~~ | | ~~Blue chips shall have a value of $25~~ |  | ~~Video~~ |  | ~~Obsoleted~~ | | Each player shall begin with $100 |  | Video | Player::wallet | Removed chips and replaced with value | | Each player shall ante $1 to enter the hand |  | Video | Game::current\_bet |  | | The ante shall establish the opening prize pot for winning the hand | Verify the initial ante is equal to the number of player processes | Video | Game::prize\_pot |  | | Each player shall check, bet, raise, call or fold after receiving their hand |  | Video | Game::check() Game::bet() Game::call() Game::raise() Game::fold() |  | | The actions shall occur in order from first player to last clockwise from left | Test that buttons are active for player in turn and inactive for players out of turn | Video | Player::reset\_sensitivity() |  | | Check means a player makes no bet ($0) |  | Video | Game::check() |  | | A player can check only if no other player has placed a bet | If prize pot > ante, check button is inactive for all players | Video | Game::check() |  | | If all players check, the prize for winning the hand is equal to the total ante |  | Video | Game::prize\_pot |  | | Bet means a player wagers some amount of his/her chips to the prize pot |  | Video | Game::bet() Player::on\_bet\_value\_changed() |  | | The first bet sets the current bet value (minimum bet amount) | Verify the bet value token after a bet is placed | Video | Game::current\_bet |  | | After a player bets, all other players still in the hand must either call, raise or fold | Set a must bet flag for all other players when a player increases the bet amount | Video | Player::has\_bet Player::reset\_sensitivity() |  | | Raise means a player wagers some amount higher than the current bet | Verify the bet value token after the bet is raised | Video | Game::raise() Player::on\_bet\_value\_changed() |  | | If a player raises, the new raised bet becomes the current bet | Set a must bet flag for all other players when a player increases the bet amount | Video | Game::current\_bet |  | | Call means a player who previously placed a lower bet than the current bet increases his/her bet to meet the current bet |  | Video | Game::call() Player::on\_bet\_value\_changed() |  | | If a player calls, the current bet value remains unchanged | Verify that the player bet amount changed to match the bet amount | Video | Game::current\_bet |  | | Fold means a player wishes to withdraw from the hand |  | Video | Game::fold() |  | | If a player folds, he/she forfeits his/her existing bet amount | Verify all buttons are inactive for folded player | Video | Player::wallet Game::prize\_pot |  | | If a player folds, he/she discards all his/her cards into the discard pile |  | Video | Dealer::exchange() |  | | After the initial betting round, players may exchange 0, 1, 2, 3 or 4 cards of their choice |  | Video | Dealer::exchange() |  | | If a player exchanges cards, his/her discards are placed in the discard pile and the dealer deals the requested number of new cards from the deck |  | Analysis | Dealer::exchange() |  | | After all players have exchanged cards or chosen to keep their existing hand, a new betting round begins |  | Video | Game::next\_stage() Game::game\_stage |  | | The second betting round follows the same rules as the first betting round |  | Video |  |  | | After the second betting round, all players hands are displayed for all to see |  | Video | Dealer::determine\_winner() |  | | The player with the winning hand earns the prize pot |  | Video | Dealer::determine\_winner |  | | A lone high card is the lowest value hand (value = 10) |  | Video | Dealer::determine\_winner() |  | | A lone high card is a card without a pair or matching suit |  | Video | Dealer::determine\_winner() |  | | One pair is the next higher value hand (value = 20) |  | Video | Dealer::determine\_winner() |  | | One pair is any single pair of cards with the same face value |  | Video | Dealer::determine\_winner() |  | | Two pairs is the next higher value hand (value = 30) |  | Video | Dealer::determine\_winner() |  | | Two pairs are any two pairs of cards with the same face values |  | Video | Dealer::determine\_winner() |  | | Three of a kind is the next higher value hand (value = 40) |  | Video | Dealer::determine\_winner() |  | | Three of a kind is any single set of three cards with the same face value |  | Video | Dealer::determine\_winner() |  | | A straight is the next higher value hand (value = 50) |  | Video | Dealer::determine\_winner() |  | | A straight is a set of all five cards with sequential face values |  | Video | Dealer::determine\_winner() |  | | A flush is the next higher value hand (value = 60) |  | Video | Dealer::determine\_winner() |  | | A flush is a set of all five cards with the same suit |  | Video | Dealer::determine\_winner() |  | | A full house is the next higher value hand (value = 70) |  | Video | Dealer::determine\_winner() |  | | A full house is a set of three cards with the same face value and a pair with the same face value |  | Video | Dealer::determine\_winner() |  | | Four of a kind is the next higher value hand (value = 80) |  | Video | Dealer::determine\_winner() |  | | Four of a kind is any single set of four cards with the same face value |  | Video | Dealer::determine\_winner() |  | | A straight flush is the next higher value hand (value = 90) |  | Video | Dealer::determine\_winner() |  | | A straight flush is a set of all five cards with the same suit and sequential face values |  | Video | Dealer::determine\_winner() |  | | A royal flush is the highest value hand (value = 100) |  | Video | Dealer::determine\_winner() |  | | A royal flush is a straight flush containing 10, J, Q, K and A from the same suit |  | Video | Dealer::determine\_winner() |  | | \*\*\*NEED TO DEFINE HOW TO HANDLE TIE\*\*\* |  | Analysis | Dealer::determine\_winner() |  | |

## User Interface

|  |  |
| --- | --- |
|  | Requirements Cross Reference   1. Button – 10.00, 10.07, 10.12, 10.13, 10.14 2. Button – (Check) 10.00, 10.02, 10.03, 10.04, (Discard) 11.00, 11.01 3. Button – (Call) 10.00, 10.10, 10.11 (Bet/Raise) 10.00, 10.05, 10.06, 10.07, 10.08, 10.09 4. Slider – 10.00, 10.05, 10.06, 10.06, 10.07, 10.08, 10.09, 10.10, 10.11 5. Client cards – 3.02, 11.00, 11.01 6. Other clients – 3.02, 13.00 7. Info – 9.01, 10.00, 11.00, 12.00, 12.01, 13.00 |

## System Models

|  |  |
| --- | --- |
|  | context diagram    state diagram  activity diagram |

# Software Design

## Class Diagram

|  |  |
| --- | --- |
|  | A screenshot of a cell phone  Description automatically generated |

## Difficult Mechanization

|  |  |
| --- | --- |
|  | The team held a lengthy discussion regarding how to structure cards. Ideally, cards could be treated as objects and passed as pointers from the deck to the player to the discard pile. However, the requirement to pass cards via json strings presents a challenge. |

# Test Procedures

## Test Case 1

|  |  |
| --- | --- |
|  | Describe first test case here |

## Test Case 2…

|  |  |
| --- | --- |
|  | Describe second test case here… |

## Defect List

|  |  |
| --- | --- |
|  | List all defects found while executing tests. Describe the defect, test case where identified, how corrected |

# Project Evaluation

## Summary

|  |  |
| --- | --- |
|  | Summarize the results of the project here |