Design Journal

for

Gin Rummy

Prepared by Devon X. Dalrymple

West Virginia University at Parkersburg

2022-03-24

Table of Contents

Table of Contents ii

1. Introduction 1

1.1 What is this design journal for? 1

1.2 How are the sections organized? 1

1.3 Updating of sections 1

1.4 Source of Rules Used 1

1.5 Other Information 2

2. Data Dictionary 2

3. The SOLID Principles 3

3.1 Why is this included 3

3.2 Sources 3

3.3 S – Single Responsibility 3

3.4 O – Open Closed 3

3.5 L – Liskov Substitution 3

3.6 I – Interface Segregation 4

3.7 D – Dependency Inversion 4

4. Use Cases 4

4.1 Brief Overview 4

4.2 Navigating to the Webpage 5

4.3 Selecting whether to do standard Gin Rummy or ‘Around the World’ Gin Rummy 8

4.4 Select the difficulty of the opponent to be played against 9

4.5 Select the number of points to play the game for 10

4.6 Start the game 11

4.7 View the rules 15

4.8 Reposition cards in their hand 16

4.9 Pass their first turn 16

4.10 Draw from the deck 17

4.11 Draw from the discard pile 18

4.12 Discard a card 18

4.13 End the turn by knocking 19

4.14 End the turn by Big Gin 21

4.15 Restart the Game 21

4.16 Quit the Game 22

5. Functional Requirements 23

5.1 Host a playable game of Gin Rummy on a Webpage 23

5.2 Allow the player to choose Standard Gin or ‘Around the World’ 23

5.3 Provide a computer-controlled opponent for the player to play against 23

5.4 Allow the player to choose a difficulty level for the game 23

5.5 Allow the player to set the number of points to play until 23

5.6 Start the game 23

5.7 Limit the beginning of each round to either draw from the discard pile or pass the turn 24

5.8 Maintain whose turn it is to play 24

5.9 Have the opponent appear to take three seconds for each decision 24

5.10 Allow the player to draw from the draw pile 24

5.11 Allow the player to draw from the discard pile 24

5.12 Allow the player to reorganize the cards in their hand 25

5.13 Show the scoreboard for five seconds after each round and for ten seconds after each game 25

5.14 Support drag-and-drop operations 25

5.15 Display the rules for the player 25

5.16 Allow the player to discard a card from their hand 25

5.17 Allow the player to knock to end the round 25

5.18 End the round due to a lack of cards 25

5.19 End the round when a player has Big Gin 26

5.20 End the game when a player has won 26

5.21 Determine the best combination of melds and points to layoff 26

5.22 Restart the game 26

5.23 Notify of an illegal move 26

6. Design Story 27

7. UI/UX 27

7.1 Color palette 27

7.2 Interaction 27

7.3 Notifying of Illegal Moves 28

7.4 Other Notes 28

8. CRC Cards (First Pass) 29

8.1 Rank 29

8.2 Suit 30

8.3 Card 32

8.4 Deck 34

8.5 Player 36

8.6 Computer Player 38

8.7 Meld 40

8.8 Meld Set 42

8.9 Meld Run 44

8.10 Discard Pile 46

8.11 Game State 48

8.12 Board 50

8.13 Knock Slot 52

8.14 Pass Button 54

8.15 Scoreboard 56

8.16 Gin Rummy Controller 58

8.17 User Interface Slot 60

8.18 Command 62

9. Design Patterns 64

9.1 Research 64

9.2 Other notes during the research process 68

9.3 MVC (mostly) 68

9.4 Singleton 71

9.5 Possible area for other design patterns 73

10. Reasons to use Blazor WebAssembly for this system 73

10.1 Blazor games are going to run faster (usually) 73

10.2 Blazor reduces stress on the Web server 74

10.3 Having parts of the page reload after each move can be jarring 74

10.4 User may not have a good Internet connection 74

10.5 Making decisions server-side will likely require some database usage 74

10.6 Previous (and similar) Web .NET games have been built with Blazor 74

10.7 What does the Web server need to do that cannot be done by the client after providing the page? 75

# Introduction

## What is this design journal for?

This design journal is meant for a Web version of the card game, Gin Rummy. It is to be built so that games can be played against an AI opponent from any popular Web browser.

## How are the sections organized?

Most sections are ordered into the chronological order that they were done during the design process. The first section is reserved for use with the introduction and the last section is reserved for the data dictionary so that they may be found in their expected locations.

## Updating of sections

As new sections of the design journal bring to light faults or previous ignorance in the design process, older sections will be updated to try to keep the whole documentation from contradicting itself. If it is found that contradictory information still exists in this document, then the newer (later) sections should be used for determining the correct information.

## Source of Rules Used

### Link

<https://cardgames.io/ginrummy/>

### Reason

This provides a playable version of the game along with rules that are well written, easy to read, and to understand. Not all rules are used as some are house rules that I felt would worsen the game to include.

## Other Information

1. Anywhere that has <S> or <H> or similar is actually the image of a shape. The images are not yet made and Unicode does not display correctly in Figma.

# Data Dictionary

1. Big Gin – When a player has drawn a card but not yet discarded a card and all 11 cards in their hand forms melds; It is worth 31 points extra
2. Deadwood – The cards and their points that are or will be harmful to the player that has them as they are not eliminated by being a part of a meld
3. Game – An entire playthrough of Gin Rummy with multiple rounds played until a score has been reached
4. Gin – When a player knocks with their hand having no remaining deadwood; It is worth 25 points extra
5. Hand – The (10-11) cards being held and interacted with by a player
6. Knock – When a player places a card face down onto the discard pile and their remaining deadwood (as the discarded card does not count) is 10 or less
7. Laying off – When a person knocks (not Gin) the opponent is able to ‘lay off’ deadwood by playing their cards off the knocker’s melds
8. Meld – A set or run
9. Rank – The letter or number found on standard playing cards (Ace to King)
10. Round – A single iteration of having cards dealt out, players playing, and being ended and scored when a person knocks or goes Big Gin
11. Run – Three or more cards that are of the same suit and increment by one from one another like (3,4,5), (Ace,2,3,4), (9,10,Jack,Queen,King), etc.
12. Set – A three or four of a kind for a rank like (4 of Hearts, 4 of Clubs, 4 of Spades); These are often referred to as rises
13. Shutout Bonus – If a player dominates their opponent using skill and/or luck and wins every round played in the game, then their points for that game are doubled as a reward
14. Suit – The shape/symbol found on a playing card (Diamonds, Hearts, Spades, Clubs)
15. Undercut – When a person knocks (not Gin) but their opponent ends up with less or equal deadwood then them during scoring, then the opponent wins the difference in addition to the undercut bonus of 25 points

# The SOLID Principles

## Why is this included

The SOLID principles are included in this design journal as to be a part of the design process from the get-go. Even though these principles would likely be still be applied to the design process, having them listed here can help to show areas of improvement related to them a lot quicker.

## Sources

1. https://www.digitalocean.com/community/conceptual\_articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design
2. https://www.baeldung.com/solid-principles

## S – Single Responsibility

The single responsibility principle pushes for classes to do one specific type of task and to do that task well. Instead of a MouseAndKeyboard class, a better following of this principle would be something like having a Mouse class and a Keyboard class.

## O – Open Closed

The open closed principle pushes for classes/objects to be open for extension but closed for modification. The public interface for methods should remain the same and results produced by these methods should not be altered in a way that causes breaking changes.

## L – Liskov Substitution

Liskov’s substitution indicates that a subclass should be able to be used in the place of a superclass without causing functionality to work in unexpected ways for the caller. A tabby cat should be able to be used in the place of a cat to call upon to meow, hiss, and such without performing erratically.

## I – Interface Segregation

The interface segregation principle states that an object/class should not have to fulfill contracts for behavior that does not relate to them. Having IRunningActor, ISwimmingActor, and IFlyingActor as interfaces is better than having a IMovingActor interface that requires the implementation of it to have details for running, swimming, and flying even if the actor cannot do one of these things.

## D – Dependency Inversion

The dependency inversion principle states that abstractions be relied upon rather than concrete implementations. By allowing the most abstract but relevant type to be used, reuse of code is improved, and as new features are added to systems, code can connect to the new but similar types.

# Use Cases

## Brief Overview

1. Navigating to the Webpage
2. Selecting whether to do standard Gin Rummy or ‘Around the World’ Gin Rummy
3. Select the difficulty of the opponent to be played against
4. Select the number of points to play the game for
5. Start the game
6. View the rules
7. Reposition cards in their hand
8. Pass their first turn
9. Draw from the deck
10. Draw from the discard pile
11. Discard a card from the hand
12. Knocking
13. Going ‘Big Gin’
14. Quitting the game
15. Restarting the game
16. **GOTO PASS OR DRAW DISCARD**

## Navigating to the Webpage

### Scenario

Actor:

1. Player

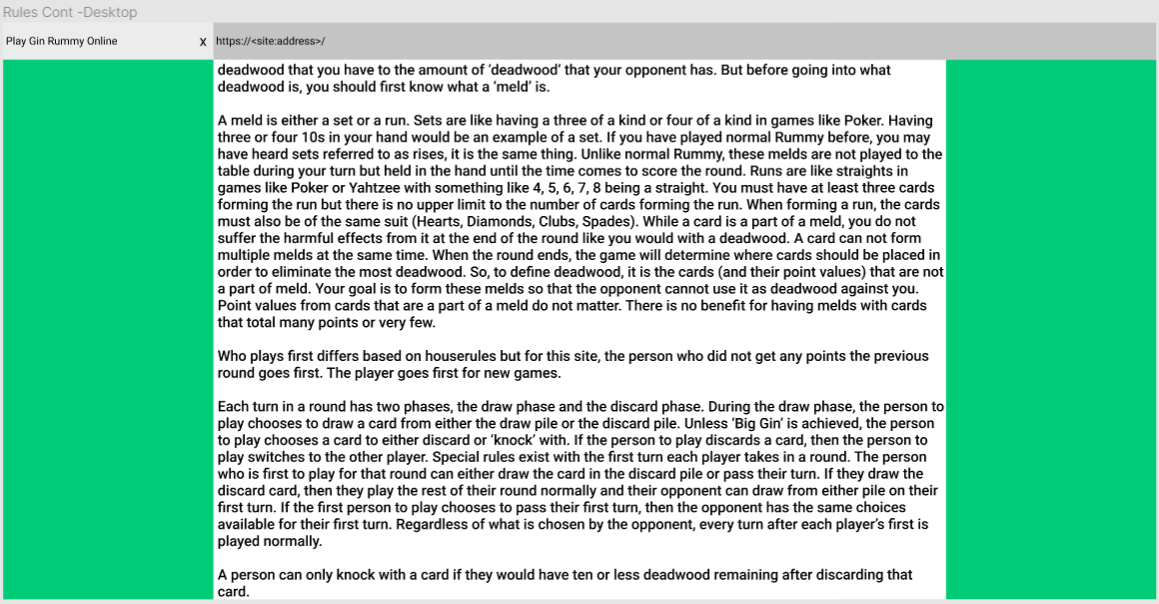
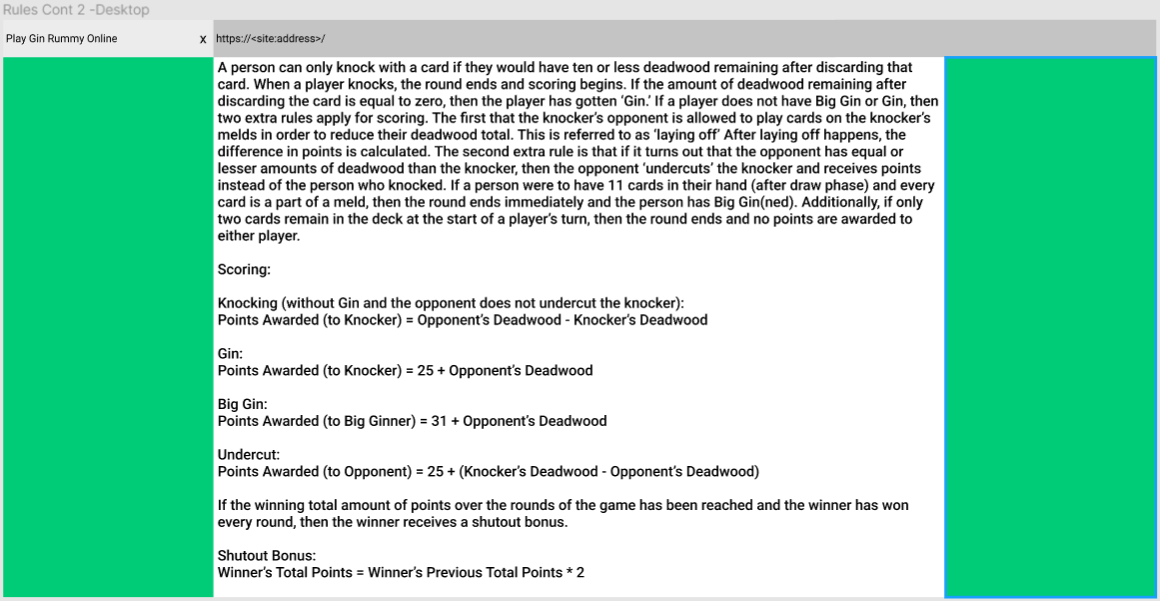
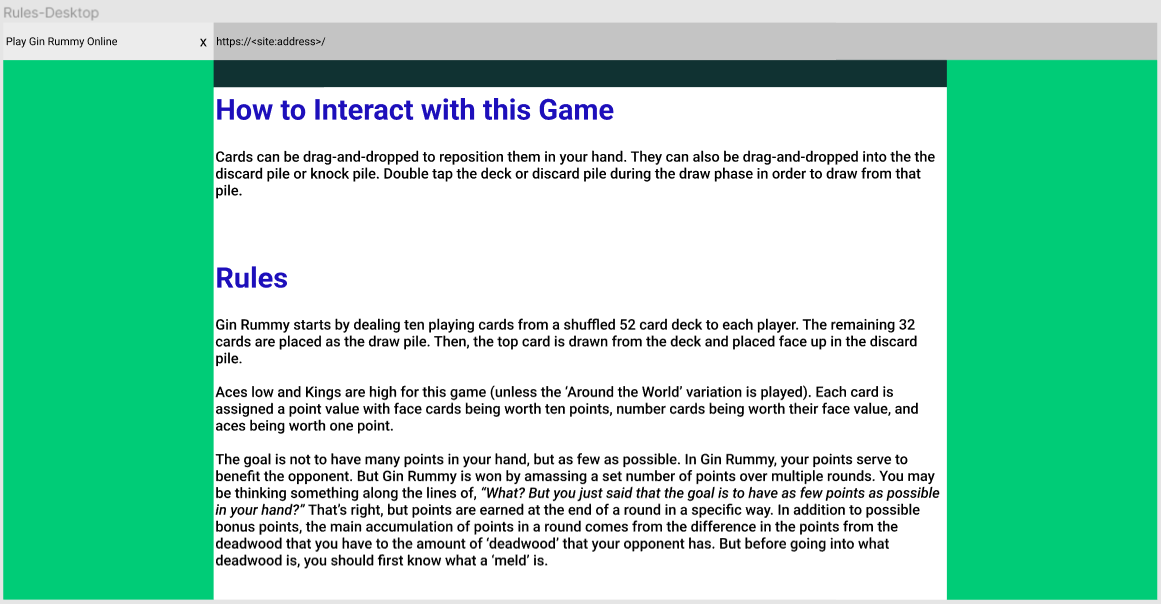
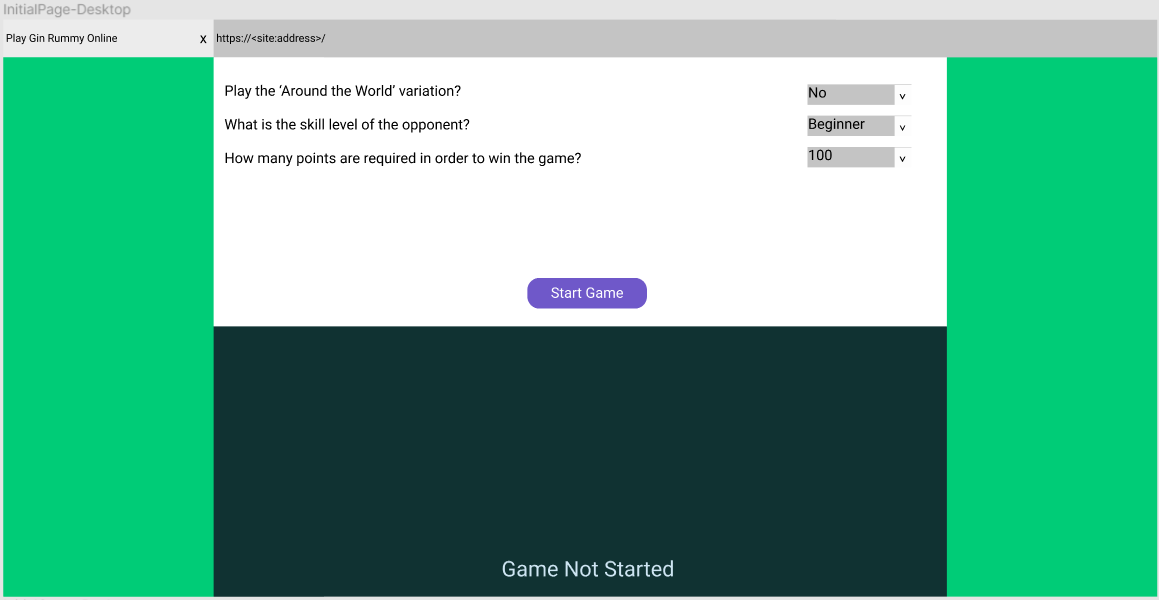
Main Success Scenario:

1. The player finds the website using their search engine or by typing in the URL
2. The player submits the URL to their Web browser or clicks on the link provided by their search engine
3. The page where the game can be played loads in front of them
4. The tab for the page says “Play Gin Rummy Online”
5. The URL they end up at is https://<site:address>/
6. If the person is on a computer or other device with a large resolution, then green borders are seen on the left and right side and the content is centered
7. The top of content contains three settings, “Play the ‘Around the World’ variation?”, “What is the skill level of the opponent?”, and “How many points are required in order to win the game?” along with dropdowns containing the default values, “No”, “Beginner”, “100” respectively
8. A purple button with “Start Game” appears below
9. A dark slate gray box appears underneath that with some centered white text stating “Game Not Started”
10. Farther down the page is a description of how to interact with the game on this page and the rules used (refer to the included design images)

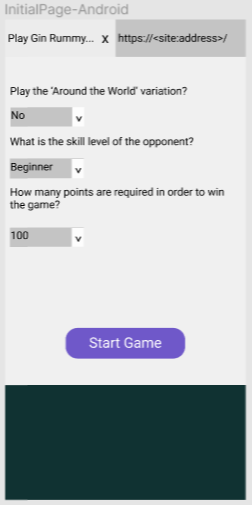
Guarantee:

1. The Webpage will load
2. Settings will be available for the player to alter
3. A start game button and box for the game appears
4. Rules and interaction instructions are at the bottom of the page

### Design Images



### Mobile View



## Selecting whether to do standard Gin Rummy or ‘Around the World’ Gin Rummy

### Scenario

Actor:

1. Player

Main Success Scenario:

1. The player clicks on the dropdown to the right or directly below of the text stating “Play the ‘Around the World’ variation?”
2. The player selects either “Yes” or “No” from the dropdown
3. The value shows as selected in the dropdown

Guarantee:

1. The value selected will show as changed
2. A new game will have this setting applied

### Conversation

|  |  |
| --- | --- |
| **User Actions** | **System Response** |
| The player clicks on the dropdown to the right or directly below of the text stating “Play the ‘Around the World’ variation?” | The system returns the available values of “Yes” and “No” and lists them in the dropdown for the player to choose |
| The player selects either “Yes” or “No” from the dropdown | The system stores the value in the dropdown for the player to see and for it to look at later on |

## Select the difficulty of the opponent to be played against

### Scenario

Actor:

1. Player

Main Success Scenario:

1. The player clicks on the dropdown to the right or directly below of the text stating “What is the skill level of the opponent?”
2. The player selects “Beginner”, “Intermediate” or “Advanced” from the dropdown
3. The value shows as selected in the dropdown

Guarantee:

1. The value selected will show as changed
2. A new game will have this setting applied

### Conversation

|  |  |
| --- | --- |
| **User Actions** | **System Response** |
| The player clicks on the dropdown to the right or directly below of the text stating “What is the skill level of the opponent?” | The system returns the available values of “Beginner”, “Intermediate” and “Advanced” and lists them in the dropdown for the player to choose |
| The player selects Beginner”, “Intermediate” or “Advanced” | The system stores the value in the dropdown for the player to see and for it to look at later on |

## Select the number of points to play the game for

### Scenario

Actor:

1. Player

Main Success Scenario:

1. The player clicks on the dropdown to the right or directly below of the text stating “How many points are required in order to win the game?”
2. The player selects “100”, “300” or “500” from the dropdown
3. The value shows as selected in the dropdown

Guarantee:

1. The value selected will show as changed
2. A new game will have this setting applied

### Conversation

|  |  |
| --- | --- |
| **User Actions** | **System Response** |
| The player clicks on the dropdown to the right or directly below of the text stating “How many points are required in order to win the game?” | The system returns the available values of “100”, “300” or “500” and lists them in the dropdown for the player to choose |
| The player selects “100”, “300” or “500” from the dropdown | The system stores the value in the dropdown for the player to see and for it to look at later on |

## Start the game

### Scenario

Actor:

1. Player

Main Success Scenario:

1. The player clicks on the “Start Game” button
2. The dark slate gray box becomes a bit lighter of a gray
3. The “Start Game” button becomes “Restart Game”
4. Ten orange gradient cards representing a face down card appear at the top on either one or two rows depending on device resolution
5. A pass button appears underneath them
6. Underneath that is the same orange gradient for a face down card slightly offset left of center with a number representing the number of cards left in the deck/draw pile
7. With the same offset but to the right and on the same line is a card face up in the discard pile
8. Underneath the deck and discard pile is a slot indicated by a purple dashed line in the shape of a card and text saying “Drag Here to Knock” to the right or underneath depending on screen resolution
9. At the bottom of the game board box is ten face up cards representing the player’s hand
10. The player is now able to play their first turn

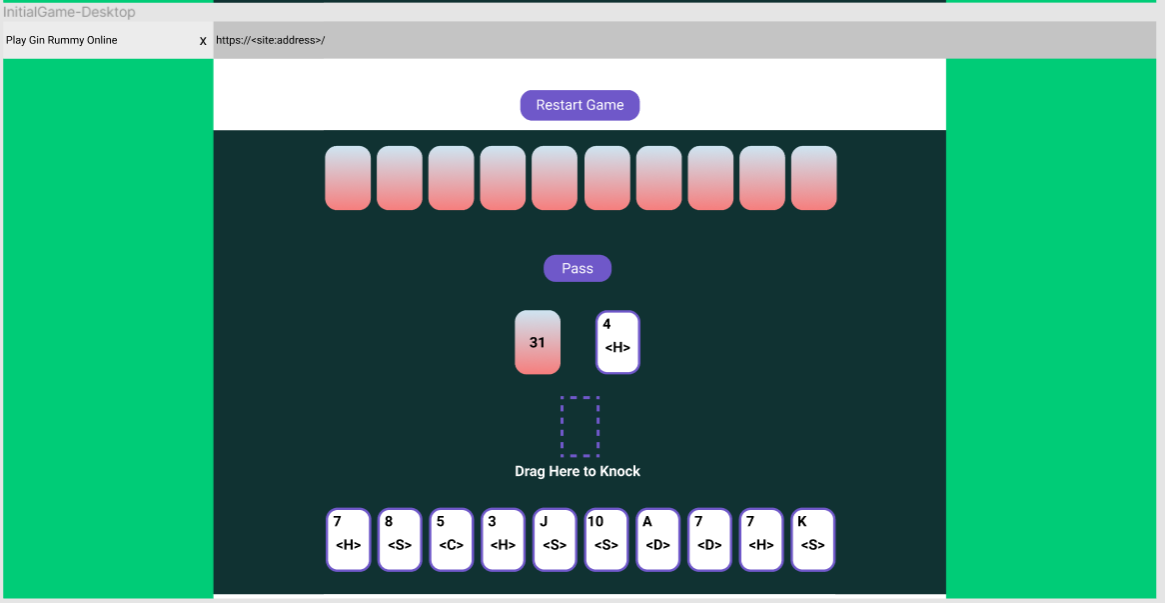
Guarantee:

1. The initial state of the board is set up
2. The player can play their first turn
3. The “Start Game” button becomes the “Restart Game” button

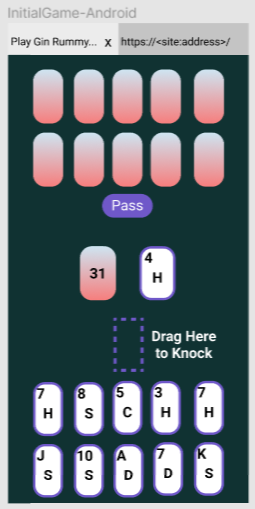
### Monologue

|  |  |
| --- | --- |
| **User Actions** | **System Response** |
| The player clicks the “Start Game” button | The system gathers all of the settings from the dropdowns and uses default values when the user has messed with the selected value in a way that the system does not know what the user is wanting  The system alters the color of the dark slate grey board box so that it is a bit lighter  The system deals out ten cards to the player and the computer  The system displays ten orange gradient cards representing a face down card appear at the top on either one or two rows depending on device resolution for the computer’s hand  The system displays a pass button appear underneath them  The system displays a gradient for a face down card slightly offset left of center with a number representing the number of cards left in the deck/draw pile  The system similarly does the same offset but to the right and on the same line with the discard pile, removing the deck’s top card and placing it face up.  The system displays underneath the deck and discard pile, a slot indicated by a purple dashed line in the shape of a card and text saying “Drag Here to Knock” to the right or underneath depending on screen resolution  The system displays at the bottom of the game board box, ten face up cards representing the player’s hand. These may be on one or two rows  The system starts waiting for the user to perform and action as the player always goes first when the game starts. |

### Desktop Setup



### Mobile Setup



## View the rules

### Scenario

Actor:

1. Player

Main Success Scenario:

1. The player scrolls down on the page past the box representing the game board
2. The rules are there underneath for the player to view (images of initial page show the rules)

Guarantee:

1. If the page loads correctly, the rules will always be visible

## Reposition cards in their hand

### Scenario

Actor:

1. Player

Preconditions:

1. The game must be running

Main Success Scenario:

1. The player starts dragging the card they wish to reposition
2. The player drags the card over to in between two other cards in their hand (or card and edge)
3. The player drops the card roughly in between the two cards (or card and edge)
4. The card is repositioned to be in between the two other cards (or card and edge)

Guarantee:

1. The player’s card is repositioned in their hand

## Pass their first turn

### Scenario

Actor:

1. Player

Preconditions:

1. The game must be running
2. The player must be the first to play or the opponent must pass their first turn
3. It must be the draw phase of the player’s turn

Main Success Scenario:

1. The player presses the “Pass” button
2. The button is removed from view
3. The dark slate grey box used for the board becomes darker to indicate that the opponent is playing

Guarantee:

1. The “Pass” button is hidden
2. It is now the opponent’s turn

## Draw from the deck

### Scenario

Actor:

1. Player

Preconditions:

1. The game must be running
2. It is not the player’s first turn or the opponent has drawn from the discard pile
3. It is the player’s draw phase of their turn

Main Success Scenario:

1. The player double clicks/taps the deck
2. The card is added to the row(s) of cards the player has, face-up
3. The number of cards left in the deck is decremented by one
4. The player can now choose a card to discard or knock with (unless Big Gin happens)

Guarantee:

1. The top card of the deck is removed and placed face up in the player’s hand
2. It is now the player’s discard phase

## Draw from the discard pile

### Scenario

Actor:

1. Player

Preconditions:

1. The game must be running
2. It is the player’s draw phase of their turn

Main Success Scenario:

1. The player double clicks/taps the discard pile
2. The card is added to the row(s) of cards the player has, face-up
3. The top card is removed from the discard pile, leaving a lower card or a slot like the one shown for knocking
4. The player can now choose a card to discard or knock with (unless Big Gin happens)

Guarantee:

1. The top card of the discard pile is removed and placed face up in the player’s hand
2. It is now the player’s discard phase

## Discard a card

### Scenario

Actor:

1. Player

Preconditions:

1. The game must be running
2. It is the player’s discard phase of their turn

Main Success Scenario:

1. The player drags the card they wish to discard to the discard pile
2. The card is added to the top of the discard pile
3. The card is no longer in the player’s hand
4. The dark slate grey box used for the board becomes darker to indicate that the opponent is playing

Guarantee:

1. The card is removed from the player’s hand and put on the top of the discard pile
2. It is now the opponent’s turn

## End the turn by knocking

### Scenario

Actor:

1. Player

Preconditions:

1. The game must be running
2. It is the player’s discard phase of their turn
3. The player would have ten or less deadwood after knocking with that card

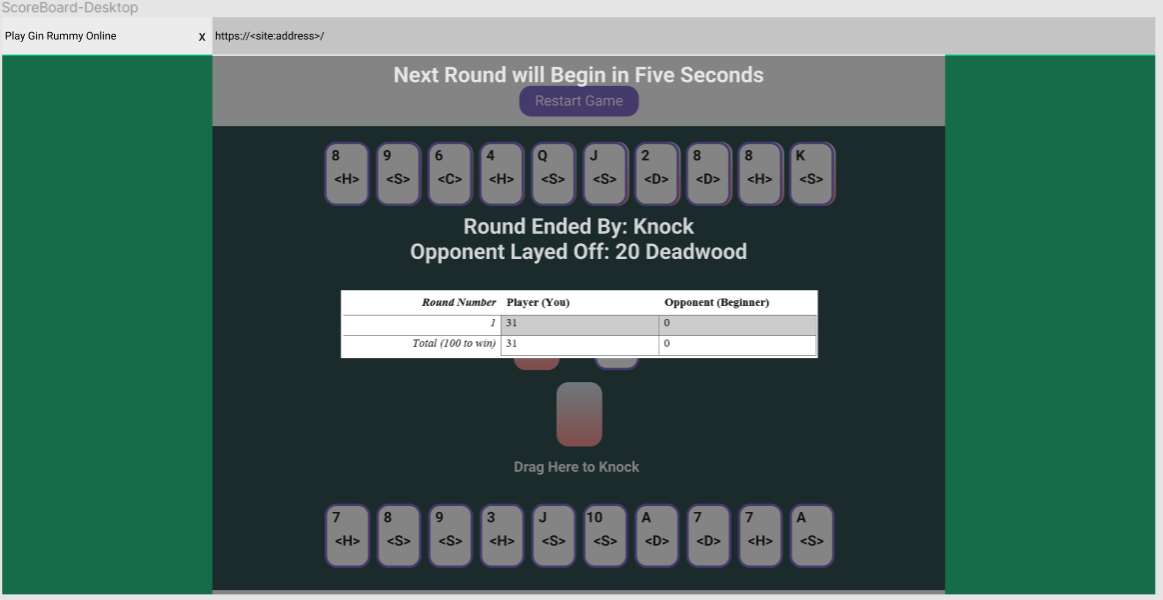
Main Success Scenario:

1. The player drags the card they wish to knock to the slot for knocking a card and drops it
2. The card is removed from the hand
3. The card is added face down in that spot
4. The opponent’s cards are shown
5. The Webpage has a transparent grey background overlay the view with information about that round and previous rounds (shown in the picture below)
   1. If it is not Gin, the laying off of the opponent’s cards happens
   2. Deadwood difference is calculated
   3. Bonus are given out
   4. Information is displayed in a table for that round and previous round
6. After five seconds, the board is reset to start a new round unless the game was won
7. Whoever did not score that round goes first in the new round

Guarantee:

1. The points are awarded
2. Opponent’s cards are shown
3. Information is displayed about that round and previous
4. A new round is set up after five seconds unless the game was won
5. The first person to play the next round is the one who did not score

### Example Scoreboard for Knocking



## End the turn by Big Gin

### Scenario

Actor:

1. Player

Trigger:

1. The player draws a card that is in a meld
2. The player has eleven cards in melds

Main Success Scenario:

1. The opponent’s cards are shown
2. The Webpage has a transparent grey background overlay the view with information about that round and previous rounds (shown in the picture below)
   1. Deadwood difference is calculated
   2. Bonus are given out
   3. Information is displayed in a table for that round and previous round
3. After five seconds, the board is reset to start a new round unless the game is won
4. If a new round starts, the opponent goes first

Guarantee:

1. The points are awarded
2. Opponent’s cards are shown
3. Information is displayed about that round and previous
4. A new round is set up after five seconds unless the game is won
5. The opponent goes first in the new round

## Restart the Game

### Scenario

Actor:

1. Player

Precondition:

1. The game must have been started

Trigger:

1. Ten seconds have past since the last game ended

Main Success Scenario:

1. A new game is set up as though a new game was started by pressing the “Start Game” button
2. Currently selected options are used to create the new game

## Quit the Game

### Scenario

Actor:

1. Player

Main Success Scenario:

1. The player navigates away from the page or closes the browser
2. Not persistence of game data effectively quits the current game

Guarantee:

1. Unless the user’s browser caches the state of game, the game state will be reset

# Functional Requirements

## Host a playable game of Gin Rummy on a Webpage

The system shall host playable Gin Rummy at its designated Web address.

## Allow the player to choose Standard Gin or ‘Around the World’

When the player expands the dropdown to the right of or directly below the text asking which version for Gin to play and selects the version, the system shall store the value and on any new games started on that browser tab will use that version of Gin.

## Provide a computer-controlled opponent for the player to play against

When a user starts a game of Gin Rummy on the Webpage, the system shall set up some form of computer player so that the user can play against someone.

## Allow the player to choose a difficulty level for the game

When the player expands the dropdown to the right of or directly below the text asking which difficulty/skill level the computer player should be and selects the level, the system shall store the value and on any new games started on that browser tab set the computer up with that skill level.

## Allow the player to set the number of points to play until

When the player expands the dropdown to the right of or directly below the text asking what number of points is required to win the game and selects the point amount, the system shall store the value and on any new games started on that browser tab set the win condition of the game to require that many points.

## Start the game

When the player clicks or taps the “Start Game” button, the system shall read the selected options and set up the board so that the player can play Gin Rummy against a computer player.

## Limit the beginning of each round to either draw from the discard pile or pass the turn

While it is the first turn for the player and first turn of the round or the opponent passed their turn, the system shall restrict the player to the choices of draw the discard pile’s card or pass their turn

## Maintain whose turn it is to play

While the game is running, the system shall maintain whose turn it is to play and what phase of their turn they are in.

## Have the opponent appear to take three seconds for each decision

This is done due to something that someone said (I think it was Almond) that due to human psychology, people like playing games against computers less when the computer seems to counter their move or makes it own play in mere nanoseconds. Also, this serves to let the player see what the computer opponent is doing better.

While it is the computer player’s turn to play, the system shall wait three seconds for both actions that the computer player takes during their turn.

## Allow the player to draw from the draw pile

While it is the player’s draw phase of their turn and special first turn move limitations are not in effect, when the player double presses the deck, the system shall remove the top card of the deck and return it to the player’s hand face up, then it should move to the player’s discard phase if Big Gin does not happen.

## Allow the player to draw from the discard pile

While it is the player’s draw phase of their turn, when the player double presses the discard pile, the system shall remove the top card of the deck and return it to the player’s hand face up, then it should move to the player’s discard phase if Big Gin does not happen.

## Allow the player to reorganize the cards in their hand

While the game is started, when the player drags and drops a card from one position of their hand into another, the system shall relocate the card to that position in the player’s hand.

## Show the scoreboard for five seconds after each round and for ten seconds after each game

While the game is either between rounds or between games, the system shall display a scoreboard for five seconds or ten seconds respectively.

## Support drag-and-drop operations

While the game is started and running, the system shall watch for drag and drop attempts from the player and handle them appropriately.

## Display the rules for the player

The system shall display the rules for the game on Webpage with the game to any users of the site.

## Allow the player to discard a card from their hand

While it is the player’s discard phase of their turn, when the player drags a card from their hand and drops it over the discard pile, the system shall remove the card from the hand in place it into the discard pile.

## Allow the player to knock to end the round

While the game is running and the player would have ten or less deadwood after discarding a card to knock with, when the player drags and drops that card from their hand into the designated knocking position, the system shall end the round and handle the knock by scoring points and awarding bonuses as necessary before starting the next round.

## End the round due to a lack of cards

While the game is running and there is two cards left in the deck, when the next draw phase for either player begins, the system shall end the round due to a lack of cards in the deck, give both players zero points for that round and start a new round with the player who went first in the ended round going first again.

## End the round when a player has Big Gin

When a player draws a card that is a part of one of their melds, while all eleven cards can be used to make melds, the system shall end the round and score points for the player in addition to the Big Gin bonus.

## End the game when a player has won

While the game is running and no current round is running, when a player has an equal or greater number of points than the win requirement, the system shall display the final scoreboard announcing the winner and restarting the game after ten seconds.

## Determine the best combination of melds and points to layoff

When a round ends, the system shall try different combinations of the cards to see the best possible melds and layoff positions that can be made and provide those points along with any bonus points to the player who won that round.

## Restart the game

When the player presses the “Restart Game” button or the last game ended ten seconds before, the system shall recreate the game board and set everything up for a new game using the currently selected options.

## Notify of an illegal move

When a player performs a move that is illegal, the system shall prevent the move and provide details through a modal dialog box about why the move failed.

# Design Story

This game playing application is being designed to be built for a Website playable version of the game, Gin Rummy. It has a number of differences from normal Rummy such as all melds staying in the player’s hand until the round ends. Unlike a recent project that used a command-line interface, this system will feature a graphical user interface. It is being made with the goal of being easy for an end user to learn how to interact with and to be visually pleasing. To try to add a visually pleasing and modern look to the game, curves are added to a lot of the shapes rather than sharp edges. An MVC architecture will likely be used for the game itself on the Webpage as prior experience with it has brought about good results. Prior projects have left behind code that can mostly be reused for this project. It should also be noted that the design of this Gin Rummy system has been heavily influenced by this online game version of it here: https://cardgames.io/ginrummy/.

An area of likely problems exists around the implementation of a computer player as no prior projects have used any proper form of a computer player so experience in that area is lacking. Although it is not guaranteed it is likely that Blazor will be used .NET 6 in order to implement this system. There are also some quality of life features are added around the three options a player can select before starting the game: the points required to win, difficulty of the computer player, and whether to play an ’Around the World’ variation where Aces can be both high and low and runs can circle around.

# UI/UX

## Color palette

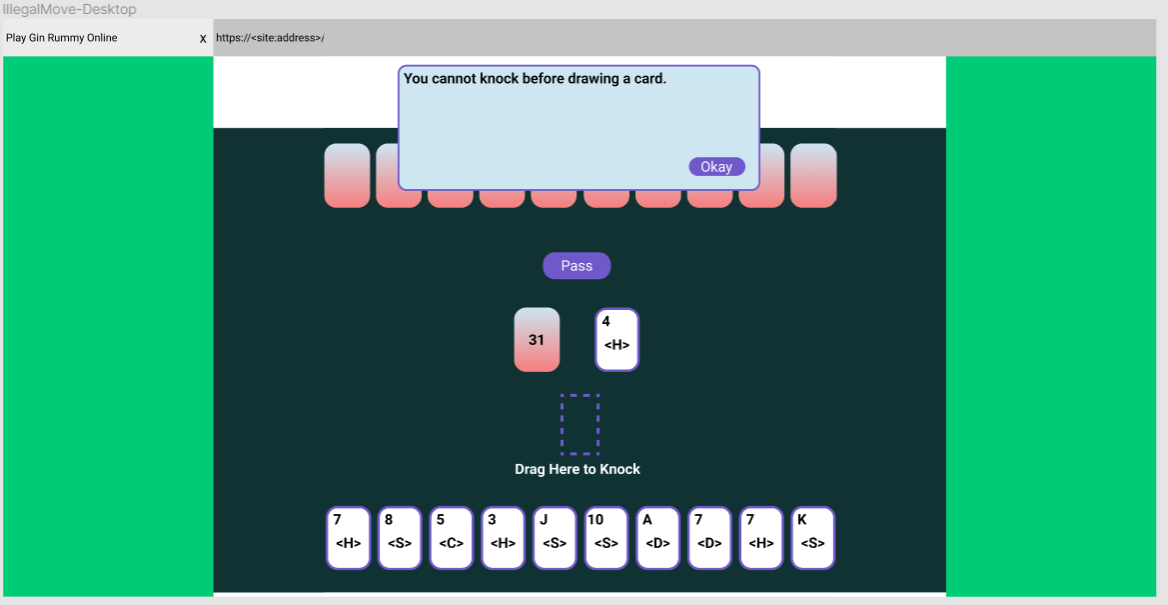
Generated using: https://coolors.co/generate



## Interaction

Interaction is handled using double taps/clicks to avoid accidental moves when dealing with the draw phase. Buttons are single press as expected by end-users. Cards can be drag and dropped to reorganize the hand in order to aid the user in playing the game. No assumptions are pushed on the player about what melds they wish to complete. The player is free to reorganize their hand however they see fit. Drag and dropped is also used for discarding and knocking. The round will also automatically end for the player if they get Big Gin because there is no good reason to continue the round at that point.

## Notifying of Illegal Moves



## Other Notes

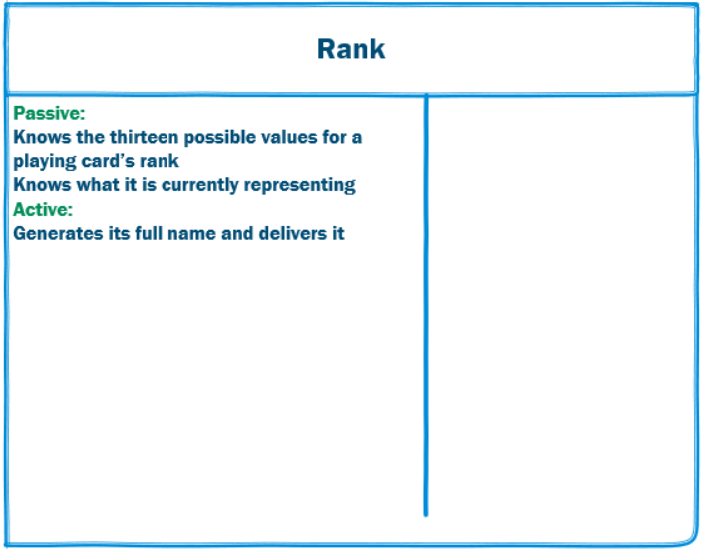
1. The deck indicates the number of cards left
2. Rounded cards appear more modern and visually pleasing
3. Designs for mobile devices were made from the beginning of the design
4. Centering the content on the screen of desktops helps the reader to not have to look left and right as much and aids with reading text on the page
5. No container margins are used for mobile due to resolution constraints
6. <S> and similar in the image represents the image of a shape; Unicode characters are not shown correctly in Figma and the card images where not ready at the time of creating these storyboard designs

# CRC Cards (First Pass)

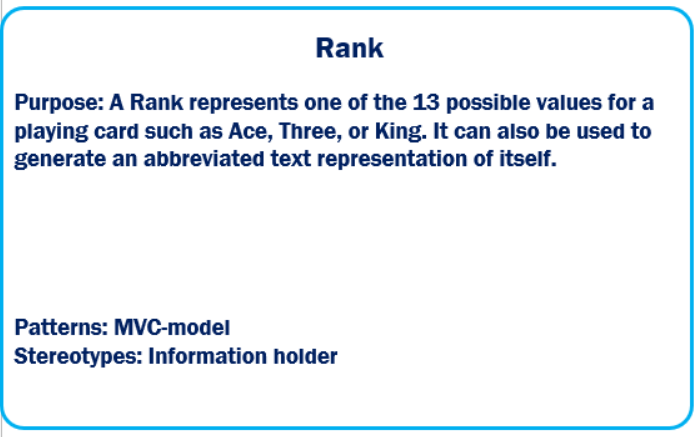
These CRC Cards will be updated after patterns are chosen and a plan is set to handle Blazor and Web display. These help to get ideas out but may vary greatly from the last pass of CRCs. Nothing is truly eliminated yet from the design.

## Rank

### Front

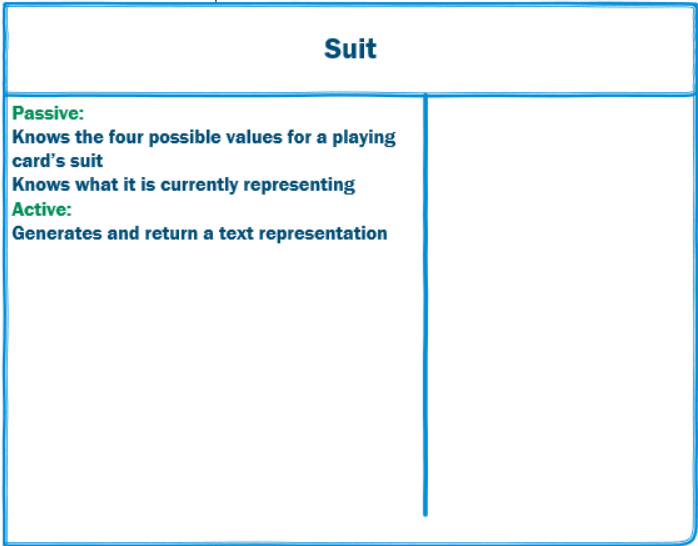


### Back

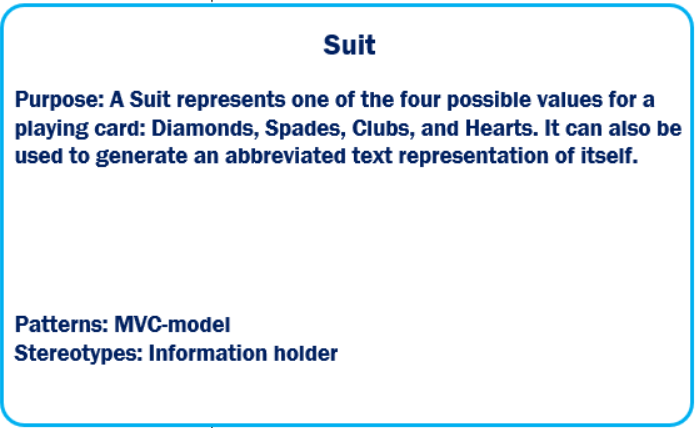


## Suit

### Front

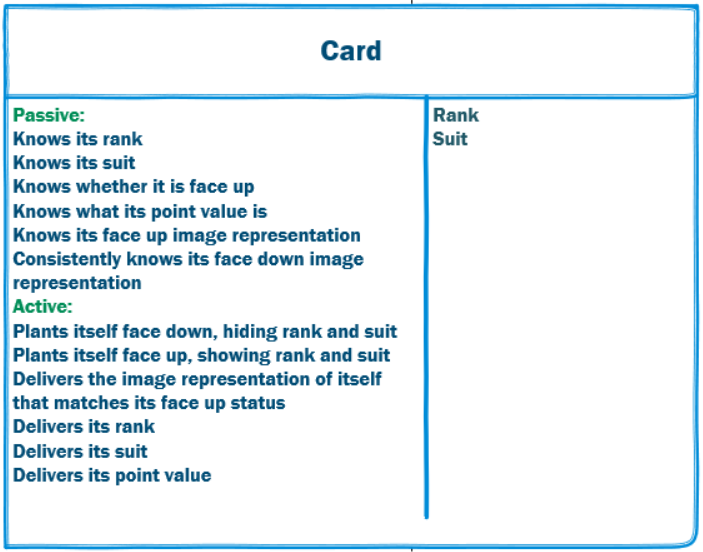


### Back

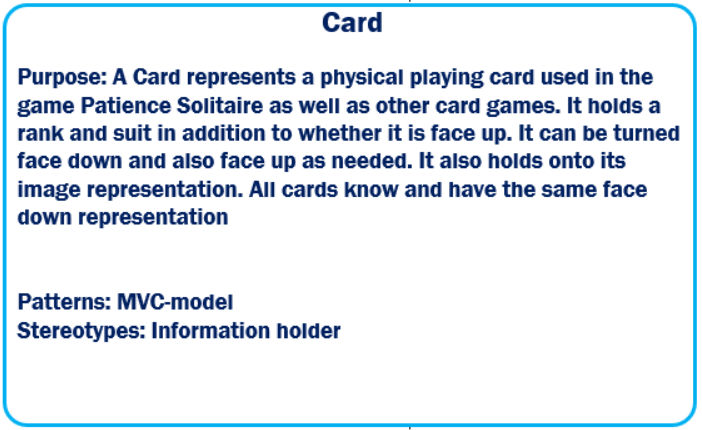


## Card

### Front

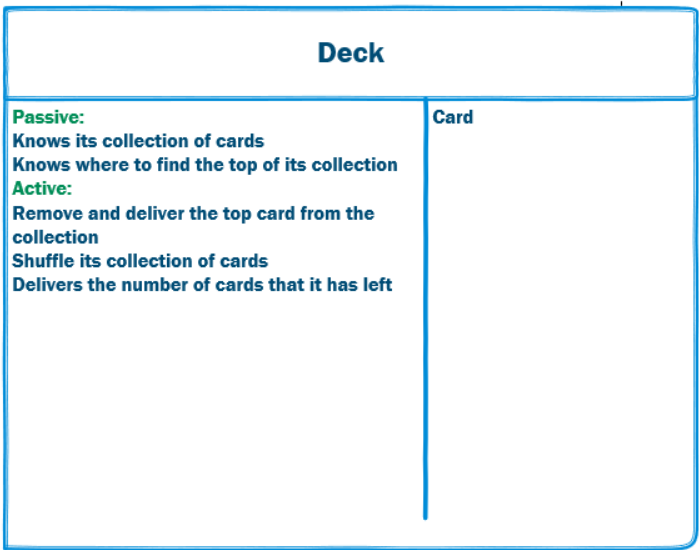


### Back

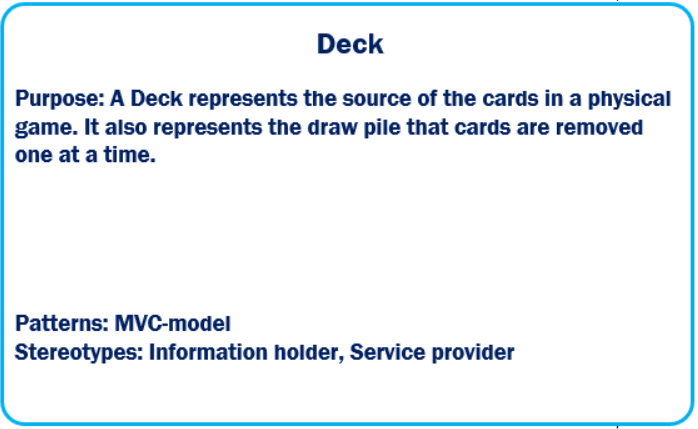


## Deck

### Front

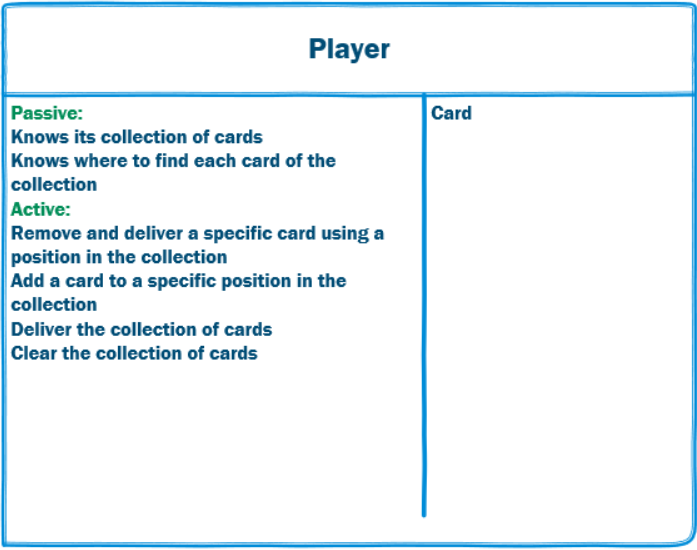


### Back

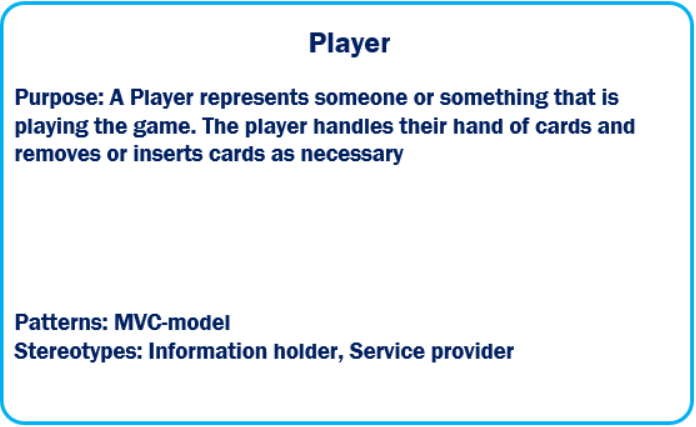


## Player

### Front

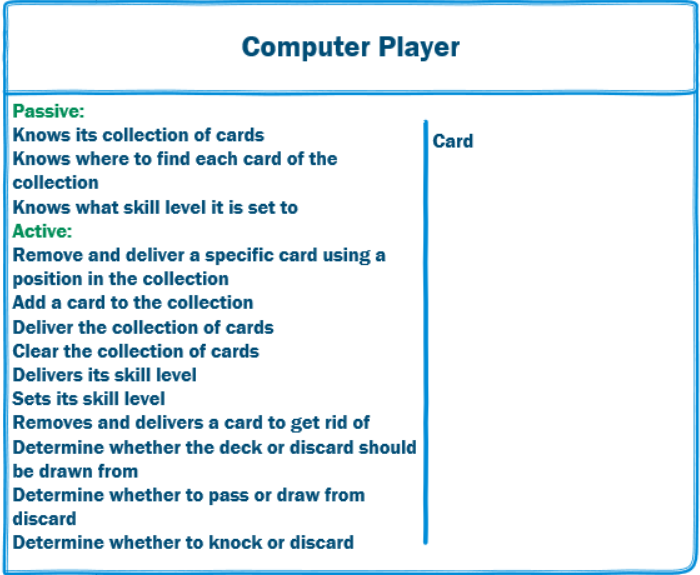


### Back

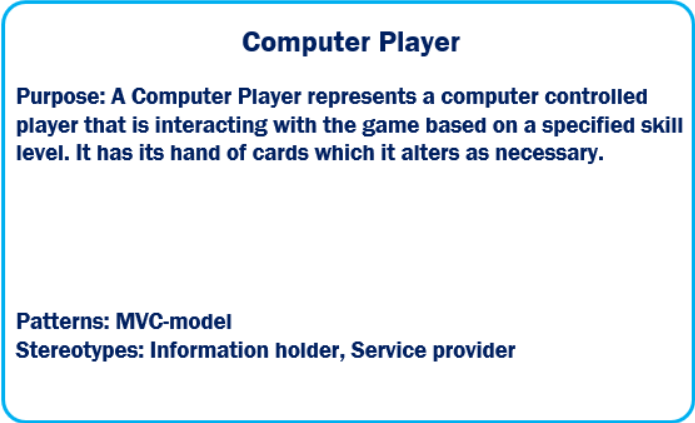


## Computer Player

### Front

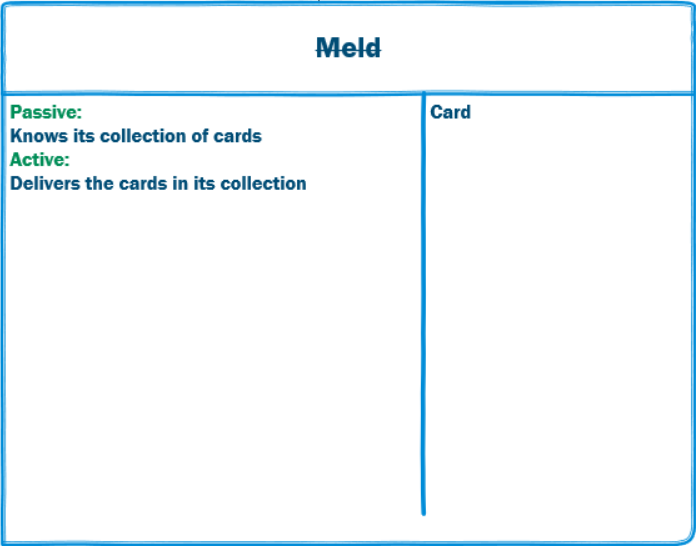


### Back

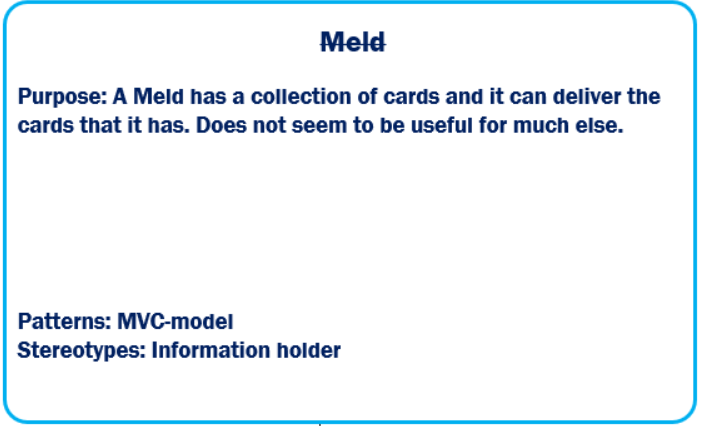


## Meld

### Front

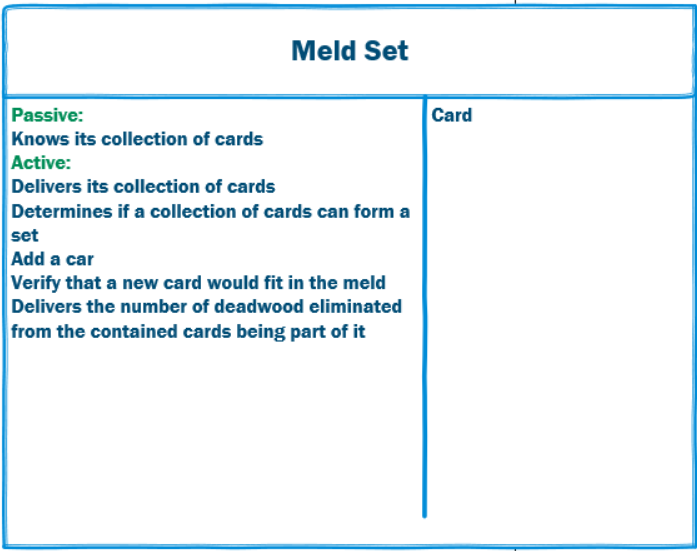


### Back



## Meld Set

### Front

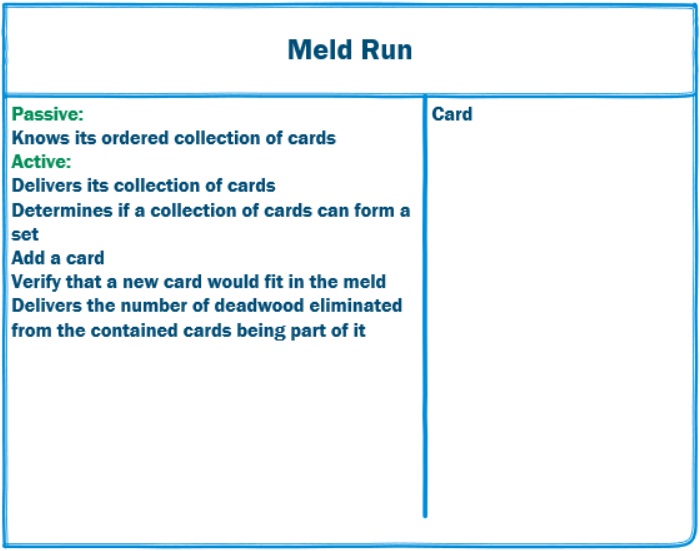


### Back



## Meld Run

### Front

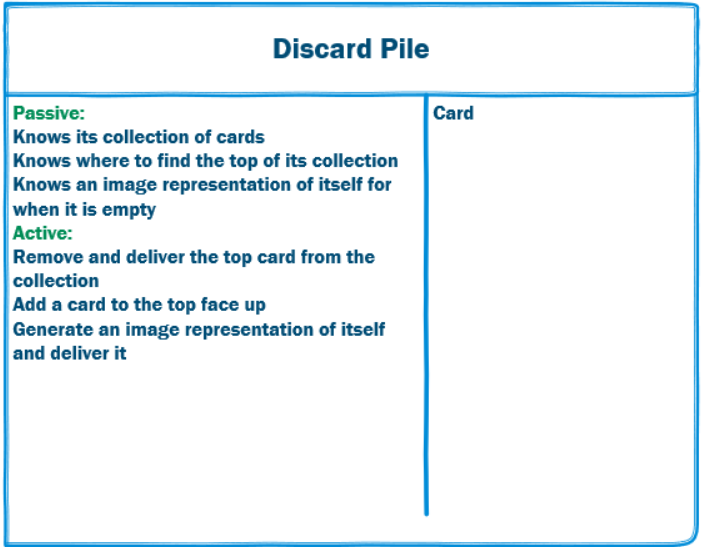


### Back

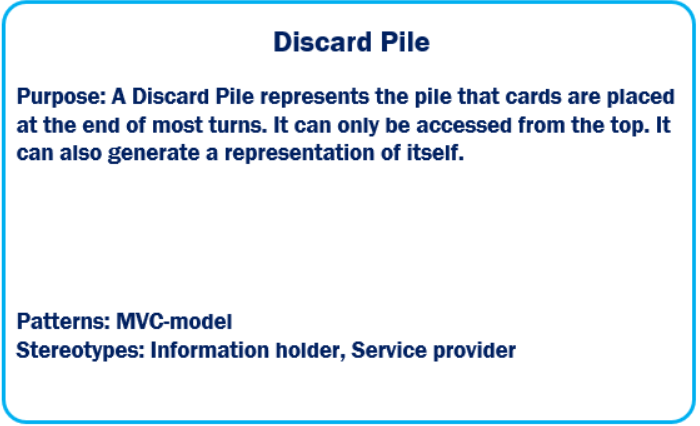


## Discard Pile

### Front

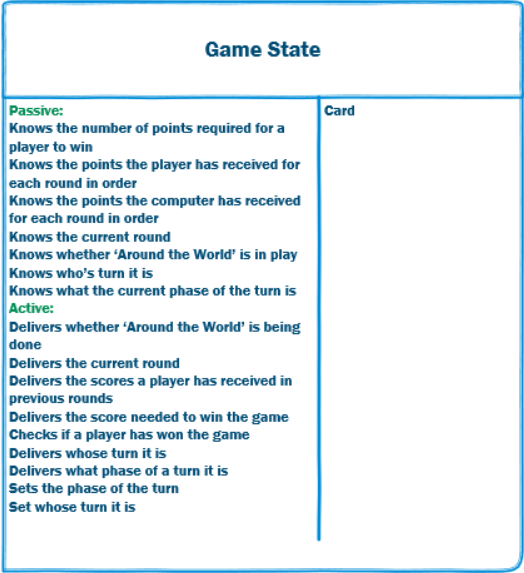


### Back

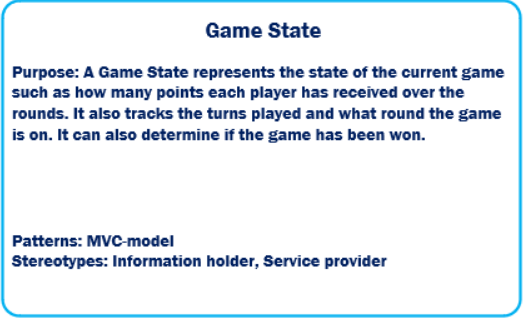


## Game State

### Front

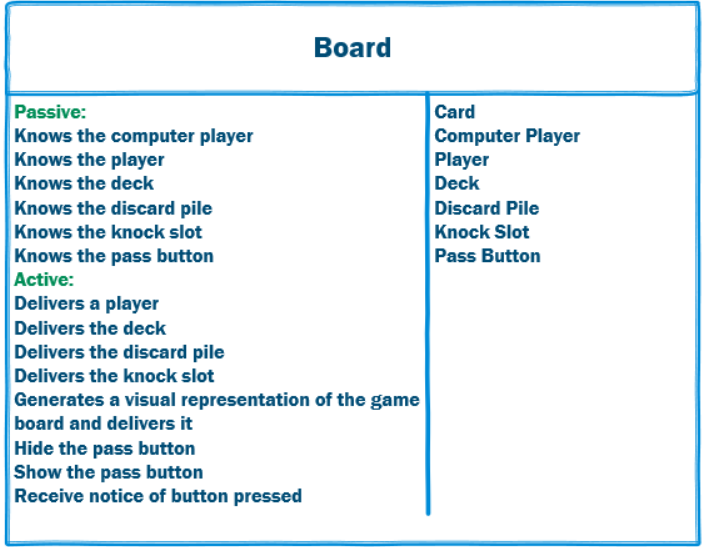


### Back

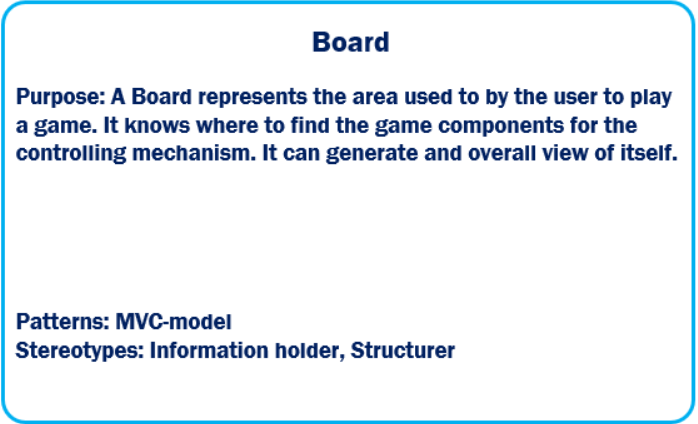


## Board

### Front

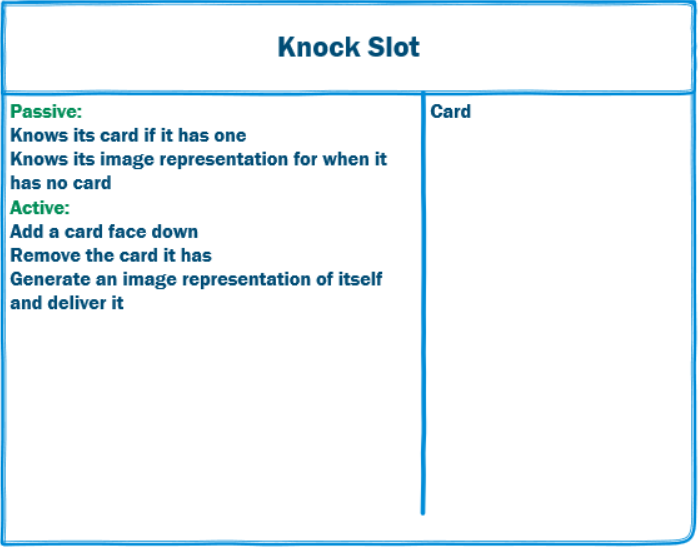


### Back

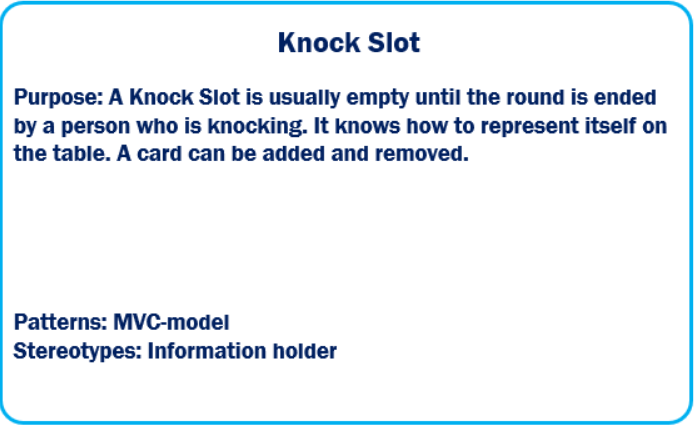


## Knock Slot

### Front

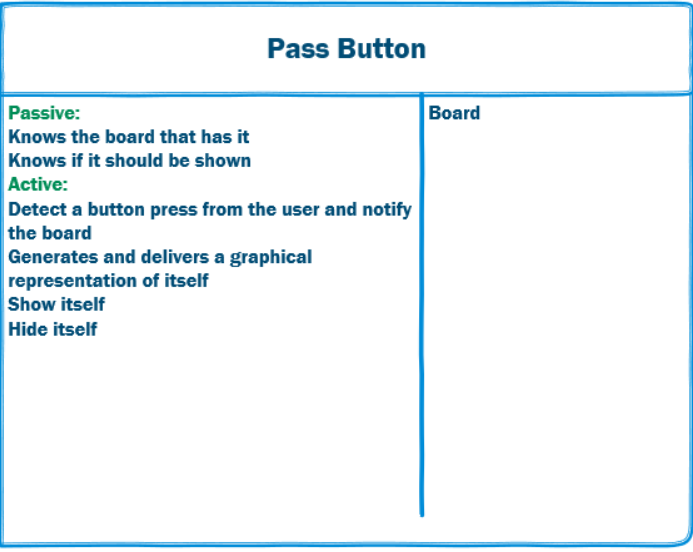


### Back

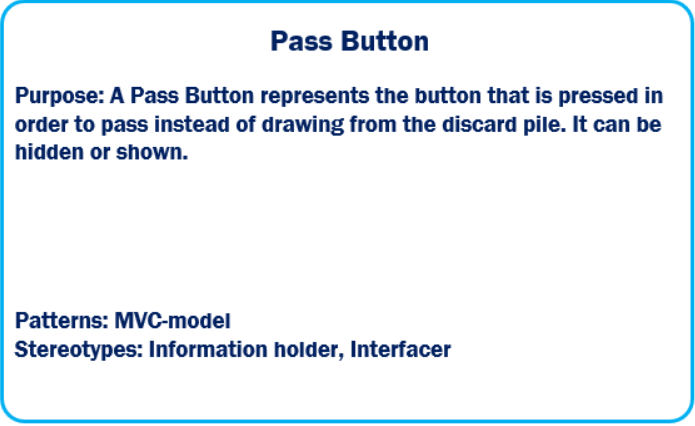


## Pass Button

### Front

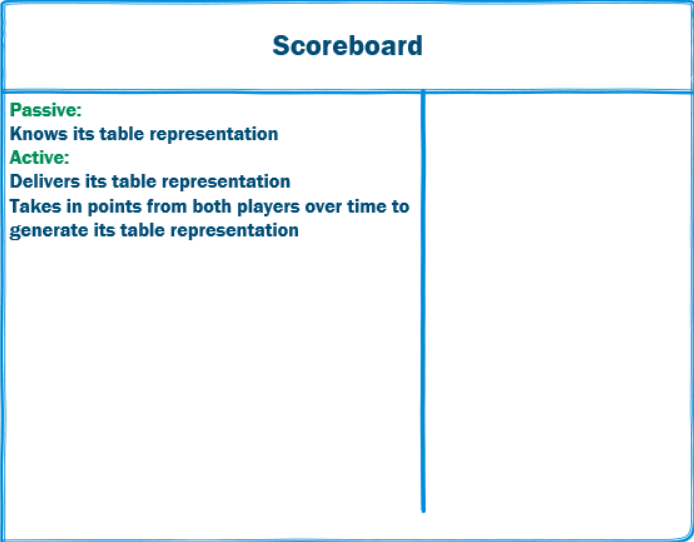


### Back

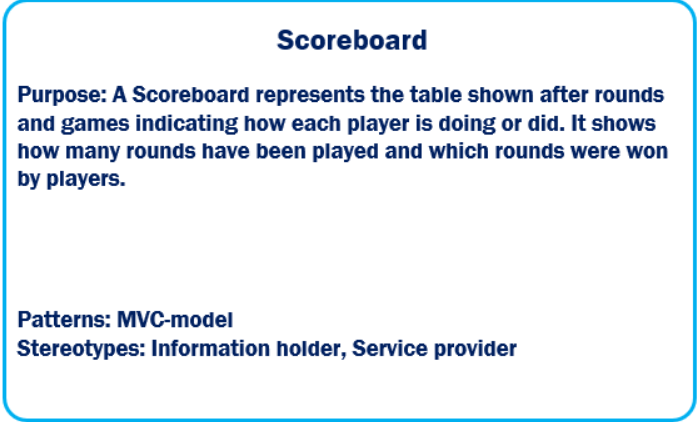


## Scoreboard

### Front

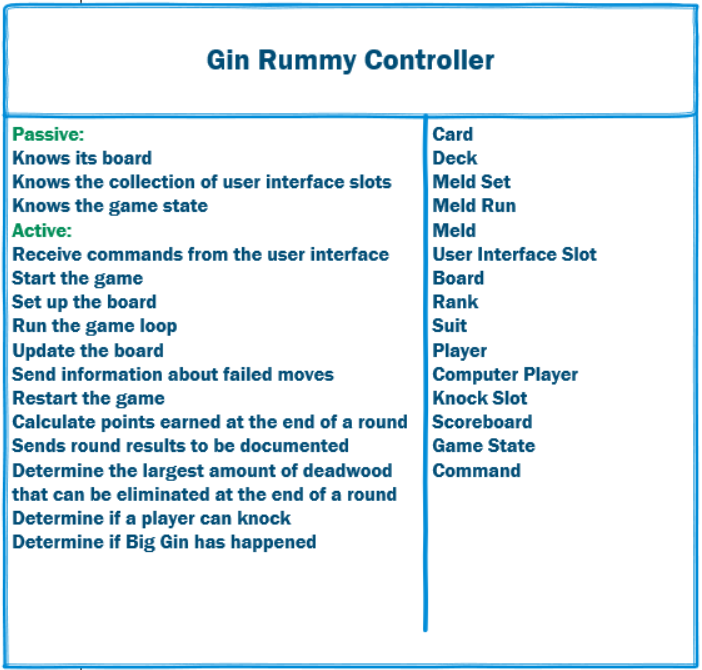


### Back



## Gin Rummy Controller

### Front

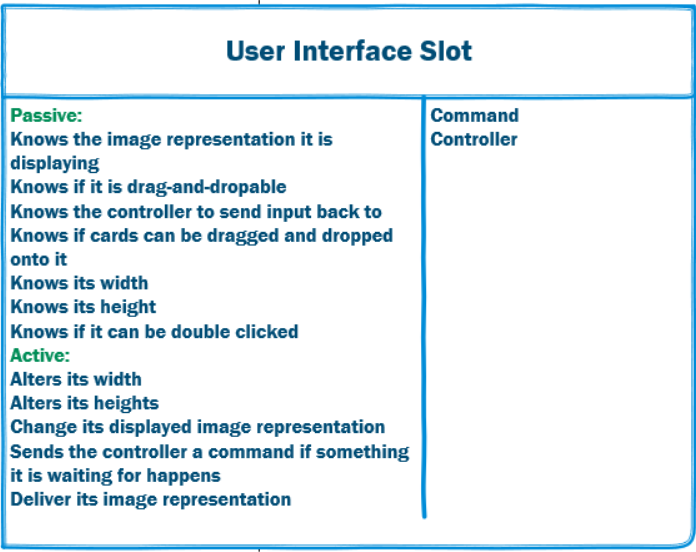


### Back

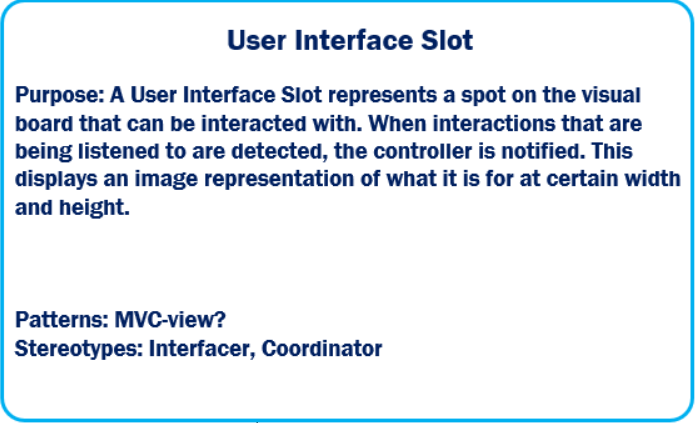


## User Interface Slot

### Front

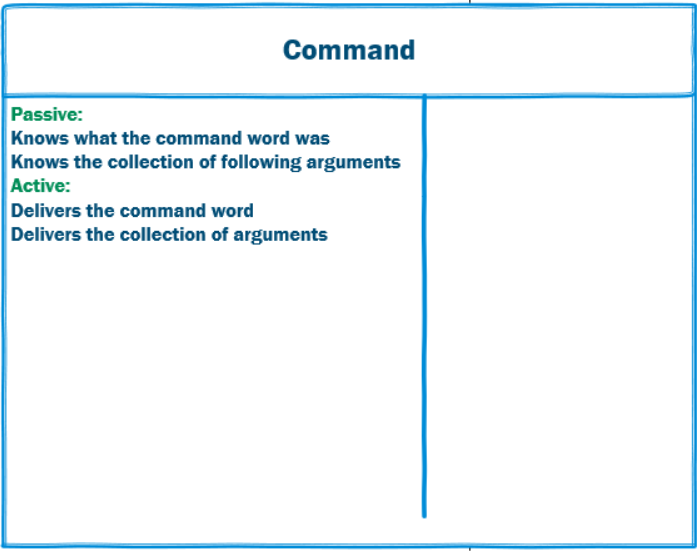


### Back

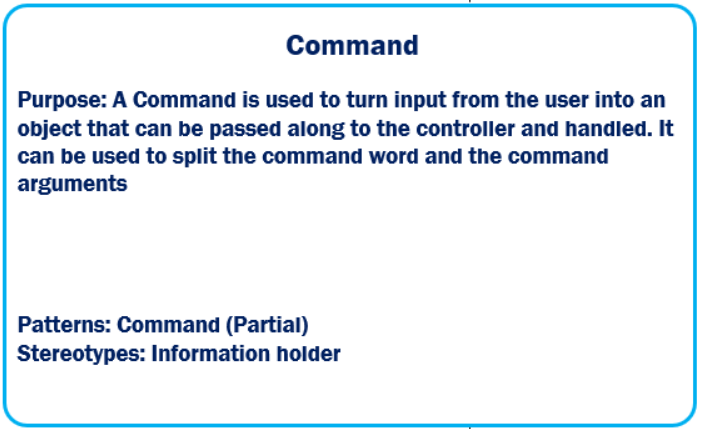


## Command

### Front



### Back



# Design Patterns

## Research

<https://github.com/exceptionnotfound/BlazorGames/blob/master/BlazorGames/Pages/Solitaire.razor>

1. Used as a resource to understand how similar projects are done
2. MIT License (Stated Here): <https://github.com/exceptionnotfound>
3. 6 examples of Blazor games

<https://www.code4projects.net/design-patterns-in-game-programming/>

1. State pattern
   1. For when an object must change based on its state
   2. Cut down on very large and complex if-else and switch statements from having many state values
   3. Like
      1. GamePaused
      2. GameReady
      3. GameRunning
2. Factory method pattern
   1. Creates objects without specifying the exact class of object to instanciate
   2. Creater class that has a factory method that uses information to determine an object to return
   3. Separate the framework from the client code allowing implementation details to differ without client modification
3. Model Control View Pattern
   1. Separate the classes that represent the data from the user interface ones and the ones that control behavior
   2. Models can remain unchanged even if the view changes and controller changes slightly
4. Game loop pattern
   1. No details provided
5. Object pool pattern
   1. Must allocate hundreds of thousands of objects during the game
   2. Objects have the same nature
   3. Limited objects needed at any given time
   4. Used a fixed number of these objects and resupply the pool after use is temporarily done
   5. Useful possibly for something like Tetris
6. Observer pattern
   1. Used when the change of state of a single object (Subject) must be notified to other objects (Observers) who show interest in it
   2. Other objects react to the notification by changing their state

<https://www.haroldserrano.com/blog/design-patterns-in-game-engine-development>

1. Singleton pattern
   1. There should only be one instance of the object that directs others
   2. The application should prevent more than one instance of this director object from being created
   3. Only one object is instantiated for a certain class
2. Strategy design pattern
   1. Games should always decouple the interaction between the input controller and the game’s logic
   2. Game’s logic should always receive the same input regardless of the type of input device
   3. Adding or removing input devices should not crash the game
3. Observer design pattern
   1. All classes should be loosely coupled in a game
   2. When a object needs to send messages to its subscribers
   3. No knowledge of how subscribers work
4. Composite design pattern
   1. For games consisting of many views
   2. Main view where characters are rendered
   3. Sub-view where player’s points are shown
   4. Sub-view showing time left in a game
   5. Each button is a view on a mobile device
   6. Each view should have a unified access point
   7. Views are placed into a tree-like structure
   8. One view is accessed and it spreads down to the other views
5. Model-view-controller
   1. Made up of three other patterns
      1. Strategy design pattern
      2. Observer design pattern
      3. Composite design pattern
   2. Controller and input source are the controller part of MVC
   3. View has subviews which are part of the composite design pattern

<https://www.gamedeveloper.com/programming/design-patterns-in-game-programming>

1. Singleton
   1. Some people see as less than a panacea than a cancer because bad architecture tends to be hidden behind it
2. Factory
   1. Use factories if other classes have complex constructors
   2. Can help with an object pool
   3. Returns a reference of the created or pooled object
   4. Declutters instantiation in the code
3. Observer
   1. When its state changes, it notifies the other objects that decided to listen to it
4. State
   1. Use an empty abstract class with possibly empty sub classes to represent states for an object
   2. PlayerInCombat, PlayerOutOfCombat, and PlayerInMenu are provided as examples

<https://www.guru99.com/mvc-vs-mvvm.html>

1. MVVM
   1. Works well HTML, CSS, and related for views
   2. Blazor is MVVM itself
   3. Instead of a controller there is a view model
   4. View model presents functions that support the state of the view
   5. Activate the events in the view
   6. Manage models
   7. View is the entry point
   8. Debugging will be more complex
   9. Code is event driven
2. MVC
   1. Will have more avenues for extensive testing
   2. Supports test driven development better
   3. SEO friendly URLs
   4. Offers to map for searchable URLs
   5. More control over the HTML
   6. Controller is the entry point
   7. One to many relationships with the controller and its views
   8. View usually does not know of the controller
   9. Reuse of this model is very difficult

<https://stackoverflow.com/questions/58551371/rendering-views-in-blazor-and-mvc>

1. Blazor can be done with MVC, it is just more difficult
2. Blazor MVC can be seen as using Blazor pages as the ‘controller’
3. The attempt is to make Blazor pattern agnostic

from: *Designing with Objects: Object-Oriented Design Patterns Explained with Stories from Harry Potter*

1. Information is being used to supplement the others for the selected design patterns

## Other notes during the research process

1. Blazor tends to be associated with MVVM

## MVC (mostly)

### Description of the general problem

MVC tries to make it so that changes in the models have no effect on the view by removing instances of coupling between them. It also tries to make it more possible for the view to swapped out with a different view as long as the other view uses the same interface by swapping out only one to a few statements at most. MVC also serves as a partial guide on where to separate functionality into different classes. MVC helps to encourage the structure of the code to be modular so parts can be altered more easily.

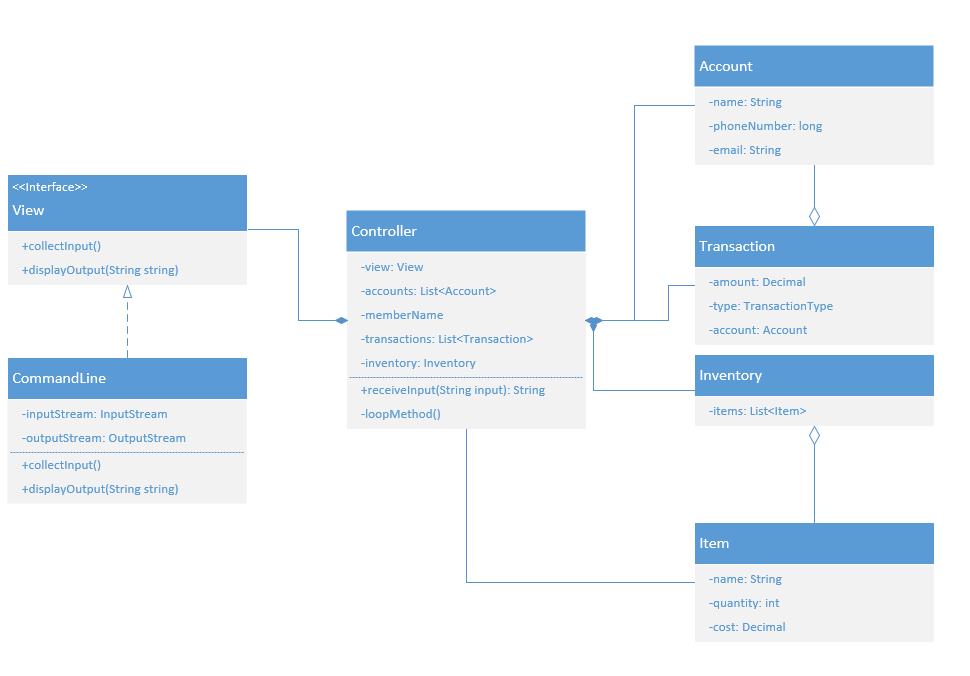
### What other games this may be used in

This design has been used in most of my previous graphical game assignments. The last two Solitaire games that I made used an MVC architectural pattern in their design. From the research done above, it is a very popular choice for games.

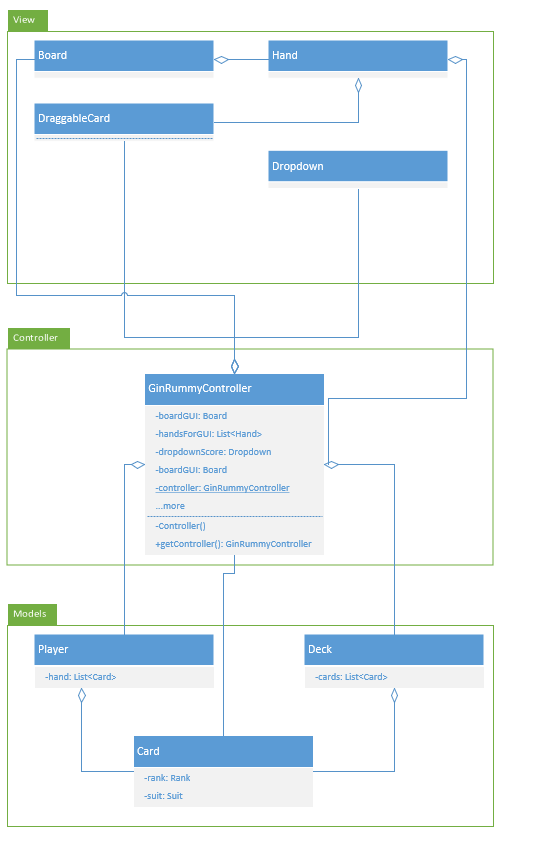
### Specific problem this solves for Gin Rummy

This may not be used to its fullest extent due to the use of Blazor components. The goal with using the MVC pattern with Gin Rummy is that it provides a centralized location that

### Explanatory class diagram



### Rough class diagram of how it will be used



## Singleton

### Description of the general problem

When there is a controlling mechanism that helps coordinate everything else it should be only created once. Having multiple of the controlling mechanism can provide large amounts of opportunity for data to fall out of sync. If two controlling mechanisms were performing asynchronous operations, it is possible that the same data might get altered twice and set to an undesired value. Associations should only ever contain one controlling mechanism object.

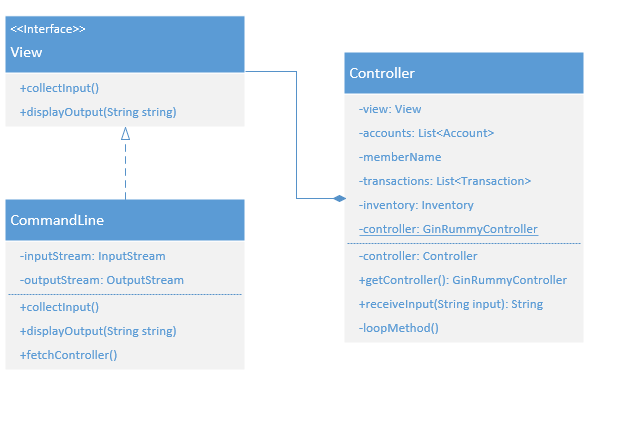
### What other games this may be used in

This may be used in many games that use the MVC pattern as the main controlling object only needs to be created once.

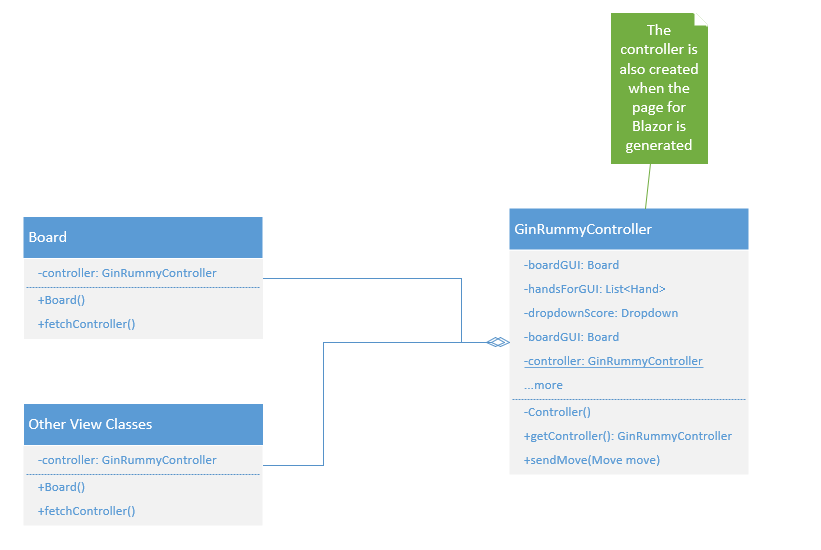
### Specific problem this solves for Gin Rummy

Since only one controller object is needed to help this system run, the controller should be a singleton. Two method calls could be done by the controller to set up the view. The first is create the view and the second is to tell the view how to talk to the controller.

### Explanatory class diagram



### Rough class diagram of how it will be used



## Possible area for other design patterns

One area that could see an introduction of design pattern or two as decisions are made is the View. The view is a group of Blazor components that will be assembled making use of composition. It would be good to get the controller to only contact one place in the views and have that properly get distributed to the other views.

# Reasons to use Blazor WebAssembly for this system

## Blazor games are going to run faster (usually)

Blazor games have their C# code run client side. While the client device is likely to be slower, this will save on time required by the passing of information between multiple places. A design using .NET MVC will likely require the system to have the client send moves to the Web server (with AJAX), have the server read the previous moves from the database, decide if the move is legal, perform any related actions, store the new state into the database, and send information to the user’s Web browser. This requires information to go to many other places. If the Internet connection is not the greatest between client and server, some responses might be lost and the game will be slower.

## Blazor reduces stress on the Web server

When the page only needs to be delivered once or occasionally, the stress on the Web server will be reduced heavily. This means a cheaper Web server can provide for more users.

## Having parts of the page reload after each move can be jarring

If AJAX or even worse no AJAX is used to handle updates of the page, then the flickering on the screen as the game updates will appear as jarring to the user. If the user makes their moves very quickly, then the flashing is even worse.

## User may not have a good Internet connection

Having the game process moves quickly using something other than Blazor will have the game handling speed rely on the Internet connection between client and server. Poor Internet connections make the game seem unresponsive.

## Making decisions server-side will likely require some database usage

Either moves will have be trusted completely and sent back from the client with other information or a database will need to be used to handle the game well. Since having information sent to the client about everything will add strain to the network, database usage will be needed.

## Previous (and similar) Web .NET games have been built with Blazor

Included in the research for design patterns is a great reference of simple games including those with cards that have been done with Blazor. Tetris with audio playback was also shown showing the feature set of Blazor is rich enough to support Gin Rummy. This reference can help deal with limited experience in Web based card games.

## What does the Web server need to do that cannot be done by the client after providing the page?

Since the game is not networked, there is nothing in particular that the server needs to do that the client cannot do themselves. Data collection of the moves the user makes is not of importance for this system.