Requirements review report

Group 2

SOFE-2720U-001

Ramiro Liscano

Due April 3rd 2020

**Table Contents**

**Introduction……….………………………………………….……...……………4**

**REQ001……….………………………………………….………………………...4**

**REQ002……….………………………………………….………………………...4**

**REQ003……….………………………………………….………………………...5**

**REQ004……….………………………………………….………………………...5**

**REQ005……….………………………………………….………………………...5**

**REQ006……….………………………………………….………………………...6**

**REQ007……….………………………………………….………………………...6**

**REQ008……….………………………………………….………………………...6**

**REQ009……….………………………………………….………………………...7**

**REQ010……….………………………………………….………………………...7**

**REQ011……….………………………………………….………………………...7**

**REQ012……….………………………………………….………………………...8**

**REQ013……….………………………………………….………………………...8**

**REQ014……….………………………………………….………………………...8**

**REQ015……….………………………………………….………………………...9**

**Conclusion…...……...………..................................................................................9**

**Introduction**

The goal of the requirements is to simply state what is required in the game. In this game Rummy-O the customer specifically stated some core requirements that must be implemented in the game. These requirements will be marked distinctively from the rest of the requirements that were created by the developers to ease the process of coding the game. Each requirement will be ordered by its requirement number. Each requirement will have the level of risk, the type of requirement it is, the scenario of a use case, and a brief description. Before any work can be done on the program the use cases should be collected and requirements should be obtained from those use cases. Next the developers, customer, and stakeholders should all have a meeting and discuss the importance of the requirements. Finally, when the requirements are reviewed the senior developer team can continue to the next step of making models of how classes and the system might look like. This is how it would work in a professional setting. However due to the nature of this project and current global evets the process is simplified.

**REQ001 Must be web application**

Risk: High Type: Functional Demanded by customer

Use case: I want to play with friends online

Brief explanation: The video game is simulating a card game. Like all card games this game is much better when played with other people rather than simple A.I. Why does it have to be online and not local? If the people you wish to play with are near by you would just play the card game in real life. Hence, the game must be online as a web application.

**REQ002 Standard deck of 54 tiles**

Risk: Low Type: Functional Demanded by customer

Use case: There has to be objects which the game is played with

Brief explanation: The game does not have any physical cards like the real life counterpart. There has to be a set of these tiles to emulate cards. Each tile has to have 2 attributes the first being an arbitrary quality, there is 4 different possible qualities. The other is an attribute that can be organized in a non-arbitrary order. Example: letters or numbers. Each primary attribute should have 13 different cards with the second attribute. Lastly there need to be 2 wildcards with neither of the previous attributes.

**REQ003 Shuffle function**

Risk: High Type: Functional Demanded by customer

Use case: Every card game starts off with a shuffle

Brief explanation: This requirement acts more as a guide. Every card game starts off with a shuffle and might require it again during gameplay. For that reason it can be made into a method that can be called when needed.

**REQ004 Mini game for choosing who starts**

Risk: Low Type: Functional Demanded by customer

Use case: Who starts the game?

Brief explanation: There has to be a way to determine the order of play. The way the customer demands is by having each player take a random card from the deck. The person that holds the card with the lowest value plays first and the order of play is clockwise from that person. This requirement is not integral to the game as the order can be determined by a randomizer but the customer wants this feature therefore it will be added to the game.

**REQ005 Method of handing out cards initially**

Risk: Medium Type: Functional Demanded by customer

Use case: How does the game start after player 1 is chosen

Brief explanation: Since cards were taken from the deck to determine who goes first they have to be placed back into the deck. Afterwards the deck is shuffled and dealt to each player. Each player receives 13 tiles and the game can finally start.

**REQ006 Has a system that allows players to make an initial meld**

Risk: Medium Type: Functional Demanded by customer

Use case: There has to be a way to start the amusing part of the game

Brief explanation: Before the initial meld is made no melds can be made by that player. A meld is either a set of 3-4 of the same cards or a run of 3 or more card with the same suit. Before the player can start making melds they must make a meld or a set of melds where the sum of every card equals or exceeds 25. After the initial meld is made by a player that player can make melds with the sum equaling to any number.

**REQ007 Game ends with a winner when a player runs out of cards**

Risk: Low Type: Functional Demanded by customer

Use case: I want to have a functioning end to the game

Brief explanations: How does the game end? A player must use up all their cards. The player does this by making melds or adding cards to melds until they run out of cards. The game has to end when one player uses up all the cards in their hand. They are promptly is declared the winner. And the round ends. The game ends when all the players leave or agree on ending the game

**REQ008 Game has a remelding function**

Risk: High Type: Functional Demanded by customer

Use case: I played Rummy once and it had this cool remelding function

Brief explanation: Once the game has started and there is a few melds on the table a player can ad to those melds or remeld. Remelding is when a player takes one or more cards from an existing meld and uses them to make a new meld from their hand. All melds that are on the table must remain valid after the remelding has occurred. Example: if a player were to take a card from a 3 set meld that would then leave that set with 2 cards which is no longer a valid meld and is not allowed.

**REQ009 Jokers have the ability to be replaced**

Risk: High Type: Functional Demanded by customer

Use case: Jokers can easily be reused and replaced in game

Brief explanation: A joker is a wild card used to substitute any card in a meld and when it is once used in a meld it can then be taken by any player to make a new meld using that joker provided that they can replace the card the joker was representing. This requirement is specified so that the developers do not find a easy way out of coding rules for the joker such as it magically becomes the card it was representing.

**REQ010 System that allows players to play in turns**

Risk: Medium Type: Functional

Use case: I want the turns to be functional

Brief explanation: This was more of a requirement used for guidance. The game must have as system that allows the players to play in turns. This can be implemented by adding a while loop to run until win = true.

**REQ11 Tiles are presented as cards**

Risk: Low Type: Interface

Use case: I want to use familiar objects

Brief explanation: In this case instead of using numbered an coloured tiles it would be simpler for the final consumer if they were playing with familiar objects in this cases cards. It also clears up any misconception amongst the developers when programming the game. Truly a win for both parties.

**REQ012 Recording of wins**

Risk: Low Type: Interface

Use case: How do I win?

Brief explanation: How does a player know who is winning and how is the winner determined at the end of the game. The real game of Rummy the winner is determined by a slightly complicated point system that would be pointless to code for this project as it does not demonstrate extra knowledge about coding and takes up more time. Instead the system for determining the winner will be round wins. First person to play all their cards wins the round. The person with the most round wins, wins the game.

**REQ013 Option to leave game when round ends**

Risk: Low Type: Functional

Use case: When can I leave the game?

Brief explanation: It is very hard to program for something where you do not have all the details and therefore have to resort to making decisions that will affect gameplay. For example, how does one leave the game? In our game it is a simple rule to follow. Once the game has started no one can leave until the round is over. When the round is over there can either be a vote to end the game or each game can have pre-set rounds. Example 3 rounds are played and the game finishes after the third round.

**REQ014 Game does not start until at least 2 people have joined**

Risk: Medium Type: Functional

Use case: I want to have the option of starting the game any time there is 2 or more people

Brief explanation: The game has to start somehow. The best way to start the game from a developer’s perspective is to allow player 1 to start the game anytime as long as there is at least one other player present in the lobby.

**REQ015 Message that notifies the players who's turn it is**

Risk: Low Type: Interface

Use case: It should be clear who's turn it is

Brief explanation: If the game is not polished enough it can be hard to know who’s turn it is. This detail should not be overlooked as it is easy to do so. For that reason, there will be a banner or some sort of message saying who’s turn it currently is.

**Conclusion**

All the requirements are stated categorized and approved. The methods of checking if the requirements work is also already set. All non-functional requirements are interface based and can be easily implemented. Everything is stated clearly and concisely. The instructions are non-ambiguous. There is not much room for error to occur because of how specific the requirements are and most of the design is left open to interpretation which allows the developers to put as much or as little work possible into the look of the game. This all depends on how fast the core game can be made. Now that the requirements are complete the developer team can go on to the next step.