COMP3211

Group Project Stage I

Software Requirement Document

of a Monopoly Game

Group 10

PAN Zewen	19******D
WANG Xiangzhi	19******D
YANG Junpeng	19******D
ZHANG Xianyi	19 ******D

Overview

Project Summary

Expected Readership: The Boss/ The Investors.

Version: 1.0

Updates: N/A

Change Rationale: N/A

Introduction

This is the Software Requirement Document of the **Command-Line Monopoly Game (CLMG)**. CLMG is a command-line version of the classic game Monopoly. This document describes the objectives and goals of the system, including the rules of the monopoly game as the functional requirement and other system requirements. In addition to describing the non-functional requirements, this document constructs functional requirements consisting of basic rules, use cases and architecture models. This document is intended to direct the design and implementation of the target system. The system architecture developed from the requirements could be changed according to future requirement changes.

Glossary

CLMG	The Command-Line Monopoly Game
CML	Command-Line
RPS	Robot Player Substitution
RRI	Related Requirement Index
KBL	Key Board Listener
MVC	The Model View Controller Architecture
OOP	Object-Oriented Programming
SR	Software Requirement
SRD	Software Requirement Document

User Requirements Definition

User Requirements

Requirement Index	**Content**
R1	Players have money and shall be able to buy properties, and both statuses of all players along with property value shall be shown in the game so that players shall check them whenever they want. Each player starts with HKD 1500 and no property.

R2	Players take turns in rolling the dice and advancing their respective tokens clockwise on the board. The order of players, remaining time of the current round, and tokens status should be clear to all players. After reaching square 20, a token moves to square 1 again.
R3	The system shall automatically make changes when players land on certain squares. Player positions and square effects shall be public to all.
R4	The system shall give choices to players and let players choose when players land on certain squares.
R5	If after taking a turn a player has a negative amount of money, she shall retire from the game. All her properties shall become unowned, and all players shall be aware of the changes.
R6	The game ends either if there is only one player left or after 100 rounds. The winner is the player with the most money at the end of the game and shall be congratulated. Ties (multiple winners) are possible. And the states should be clear to all players.
R7	Players shall be able to create, save, and load games.

Non-Functional System Requirements

Requirement Index	Content
R8	The monopoly game shall command-line line game.
R9	The game shall be played with a board divided into 20 squares.
R10	The dices used shall be four-sided (tetrahedral) dices and there are two of them.
R11	The game shall accommodate 2-6 players.
R12	If a player in an ongoing game is going to leave, the player should choose either Quit (Cannot back to the game) or Robot Player Substitution (**RPS**) (Can back to the game).
R13	All players shall start from the first square (“Go”).
R14	A round shall consist of all players taking their turns once.
R15	The order, price, rent of the properties in the Game Board shall be able to be changed. The number of properties shall not be changed, as well as squares Go, In Jail, Chance, Free Parking, and Go to Jail.

System Architecture

Background

Model-View-Controller (MVC) Architecture shall be implemented in this system. It is based on the idea that an application is consist of three main logical components: the model, the view, and the controller.

Architecture Requirements

1. Easy to Division:

   Many programs/Software/Games can be divided into Model, View, and Controller because the computer is designed based on the (Interface and I/O), (Core computation procedures), and (Status manager or Controller). By doing so, the structure of the whole game becomes very clear and easy to distinguish. Also, the benefit of easy division is that the probability of error is lower.

2. Development Friendly:

   This is more conducive to collaboration development. We will not waste time arguing the structure of the game and division of labour. The things we need to do are to take the communication among the modules into a standard agreement where the UML diagram will be used and make the game run and bug-free.

3. Development Experience:

We have implemented MVC architecture last year since we did the OOP Group Project. As a result, we can prioritize more easily and work more efficiently. In addition, we will use MVC in our future study and work, so this project can also be used to accumulate experience and improve ability.

The Functions

1. View

View is about the Interface and I/O.

- Key Board Listener (KBL). Listen to the user's input whenever needed and check the validness of the current input (inputs of the game is fixed.)
- The Game Board (GB). The Interface when the game is playing. Including the game board itself, the status of every user (where they are, the properties they have, the money they have, and their current position.), the statues of the whole game (who's round, which round)
- Menu. The game Menu including play, quit, settings, manual selections. Also, the Menu can be used for selector if override the "buttons" of Menu.
- Message Box. A CML version Message Box applying (yes, no), (ok), etc. Also, support for an override.
- Dice. Random dice.

- Manual. The pre-edited user manual.
- Other Interfaces. Like welcome interface.

2. Controller

The manager of the whole game. The controller may control where the next step to go after the specified user operation or timer callback of the Controller itself, the status will change to next. E.g., when the players select the play button on the Menu, the controller makes the game into playing, call model to initialize the data of players, call interface to load the GB, and call the keyboard listener to wait for the specific user input.

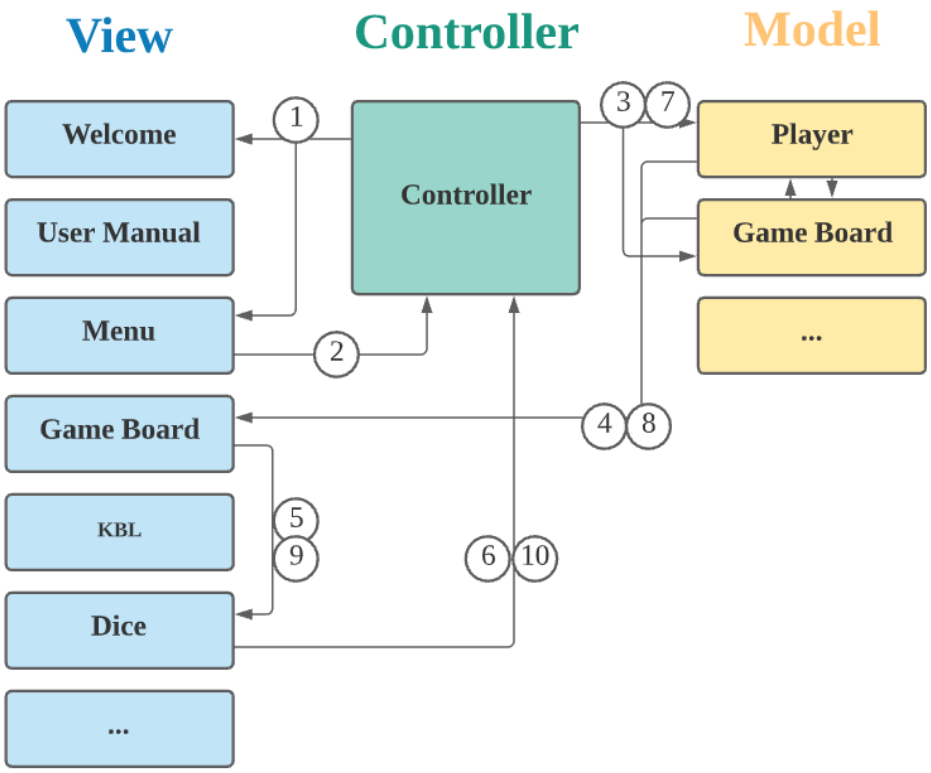
3. Model

The back end, computation core, and recorder of the game.

- Saver & Loader. Can recover the saved game when opening. Can save the game when quitting.
- Player-Object. Store the data of the player during the whole game procedure.
- Game Board. Backend game board to store the game data (player location, round num, player num, player names, etc.) indicate the next step to go.
- Robot Player. Based on a random algorithm or greedy algorithm or others. Allowing the game to support single-player mode and possibly player instead of human players.

The Interaction Mechanism

Once the game is started, the Controller enters an infinite loop to make the View listen to the user's inputs. The View will tell the controller the next step like play the game, and the Controller will call the Model if necessary and make View listen to the next input. i.e., Once started the Menu is called by the controller and after the user enters a valid input the controller enters the next state based on the input. If the input is playing the game, then the Controller will make the Model load the game and call the interface switch to the Game Board page, then the game begins.



Here is a simple diagram above to demonstrate the interaction mechanism from step 1 to step 10 once the game is started. (Some parts of functions and procedures are omitted.)

System Requirements Specification

Functional Requirements

Title	RRI	Content	Priority
state	R1	Users can see their current state (amount of money, properties ownership, which players are retired, remaining time of current round and so on, and the order of operation). The initial amount of money is HKD 1500.	1
endGame	R6	When a player has a negative amount of money, he is out of the game. When there is only one player left, the left one is then the winner. If after 100 rounds there are still multiple players left, the winner is then the players with the most money at the end of the game. Ties (multiple winners) are possible.	1
pause		During the game, local players can pause a game whenever they want.	1
Move Player	R2	A player moves based on the dice roll (with four faces). In multi-player game, if a player doesn't roll the dice in 30 seconds, the system should automatically roll the dice to make the game	1

		continue. When the user reaches the end of the board, he cycles around.	
Move Players in Turns	R2	The players should play in turns	1
passGo	R3	When a player passes or lands on the GO cell, the bank gives the player HKD 1500 salary.	1
freeParking	R3	When a player lands on Free Parking, nothing happens.	2
tax	R3	When a player lands on squares which are owned by the same player, the player loses 10% of the money (rounded down to a multiple of 10) as income tax.	2
justVisiting	R3	When a player lands on in jail/Just Visiting not by landing on “Go to Jail”, nothing happens. Details shall be found in Appendix.	2
jail1	R3	When a user lands on the "Go to Jail" cell, the player goes directly to jail (the in jail square), does not pass go, and does not collect HKD 1500.	2
jail2	R4	When a player is in jail, she may choose to throw doubles (i.e., both dice coming out the	2

		same face up) on any of her next three turns (if she succeeds in doing this, she immediately moves forward by the number of spaces shown by her doubles throw) to get out of jail in the next turn. If a player doesn't roll the dice in 30 seconds, the system will automatically roll the dice to make the game continue.	
jail3	R4	When a player is in jail, she may choose to pay a fine of HKD 150 before she rolls the dice on either of her next two turns. If the player does not throw doubles by her third turn, she must pay the HKD 150 fine. She then gets out of jail and immediately moves forward the number of spaces shown by her throw. If the player doesn't have enough money, she can only choose jail2's method to get out.	2
purchasingProperty1	R4	When a player lands on a property cell, and it is available, the player may purchase it. If a player doesn't decide in 30 seconds, the system will automatically give up the chance to make the game continue. The price is the land value which is mentioned in	1

		Appendix A.	
payRent1	R3	When a player (A) lands on a property owned by another player (B), A must pay rent to B. The rent is mentioned in Appendix B.	3
payRent2	R5	If player B owes player A more money than player B currently has, player B is bankrupt and out of the game. He must give all his money to player A and all the properties he owns become available.	3
chance1	R5	When a player lands on a Chance cell, the system starts a random number from -300 to 200 (multiple of 10). If the number is negative, the player pays the money to the bank. If he does not have enough money, he is out of the game, and the cells he owns become available without any houses.	2
chance2	R3	When a player lands on a Chance cell, the system starts a random number from -300 to 200 (multiple of 10). If the number is positive, the player gets the money from the bank.	2

Non-functional system requirements

(**Note:** The **R#Num-#Statement** means that the statement is corresponding to the requirement in the section **User Requirement Definition**.)

1. Reliability
   - The system should be operational whenever users want to use.
   - The mean time for the system to respond to user operations should be less than 1 second.
2. Usability
   - A user should be able to use the system immediately after reading the user manual.
   - A user who already knows monopoly game rules should have no difficulty using the system.
   - R7- Users can create a new game or load the saved game. During one game, users can pause and save whenever they wants.
   - R12-If a human player in an ongoing game is going to leaving, the player may choose either Quit (Cannot back to the game) or Robot Player Substitution (**RPS**) (Can back to the game).
3. Performance
   - R11-The system should be able to support 2-6 users in one game.
   - The mean time for the system to show the status of the players should be less than 3 seconds.
4. Supportability

- R8- The monopoly game is a command-line game.

5. Correctness
   - R10- The dice used is four-sided (tetrahedral) dice.
   - R13- All players start from the first square (“Go”).
   - R14- A round consists of all players taking their turns once.
6. Interface

   The system’s interface is a board with 20 squares including several functional squares. The functional squares can be changed in different games. Below is a sample game board not representing the final version.
   - R9- The game is played with a board divided into 20 squares.
   - R15- The order, price, rent of the properties in the Game Board can be changed. The number of properties cannot be changed. The squares go, In Jail, Chance, Free Parking, and Go to Jail cannot be changed.
7. Maintainability
   - Java to develop the game.
   - MySQL to store the game data.

Appendices

Appendix A: The rules of the monopoly game

The game is played with a board divided into 20 squares and a pair of four-sided (tetrahedral) dice and it can accommodate two to six players. Besides playing the game, Players should also be able to save and load a game.

It works as follows:

- Players have money and can own properties. Each player starts with HKD 1500 and no property.
- All players start from the first square (“Go”).
- Players take turns in rolling the dice and advancing their respective tokens clockwise on the board. After reaching square 20, a token moves to square 1 again.
- Certain squares take effect on a player (see below) when her token passes or lands on the square. For example, they can change the player’s amount of money.
- If after taking a turn a player has a negative amount of money, she retires from the game. All her properties become unowned.
- A round consists of all players taking their turns once.
- The game ends either if there is only one player left or after 100 rounds. The winner is the player with the most money at the end of the game.

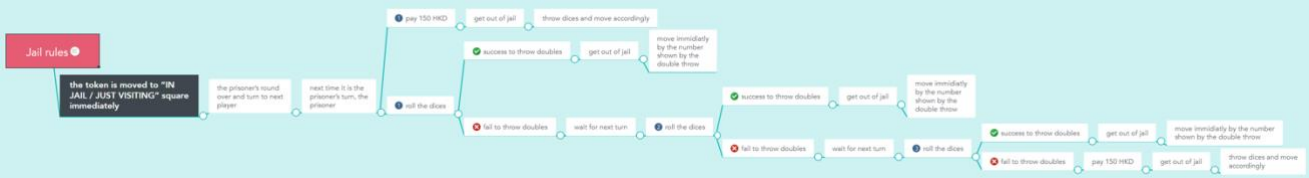
Ties (multiple winners) are possible.

There are seven kinds of squares on the board:

- Property squares (marked by a colored stripe). They contain the name and the price of the property and can be owned by players. If a player lands on an unowned property, she can choose to buy it for the written price or do nothing. If a player lands on a property owned by another player, she must pay rent.
- Go. Every time a player passes through (not necessarily lands on) this square, she gets HKD 1500 salary.
- Chance. If a player lands on one of these squares, she either gains a random amount (multiple of 10) up to HKD 200 or loses a random amount (multiple of 10) up to HKD 300.
- Income tax. If a player lands on this square, she pays 10% of her money (rounded down to a multiple of 10) as tax.
- Free parking. This square has no effect.
- Go to Jail. If a player lands on this square, she immediately goes to the “In Jail” part of the “In Jail/Just Visiting” square.
- In Jail/Just Visiting. If a player lands on this square, she is “Just Visiting”: the square has no effect. However, if the player got here by landing on “Go to Jail”, she is in jail and cannot make a move. A player gets out of jail by either throwing doubles (i.e., both dice coming out the same face

up) on any of her next three turns (if she succeeds in doing this, she immediately moves forward by the number of spaces shown by her doubles throw) or paying a fine of HKD 150 before she rolls the dice on either of her next two turns. If the player does not throw doubles by her third turn, she must pay the HKD 150 fine. She then gets out of jail and immediately moves forward the number of spaces shown by her throw.

- Details related to Jail squares:



Appendix B: Properties

Position	Name	Price	Rent
2	Central	800	90
3	Wan Chai	700	65
5	Stanley	600	60
7	Shek O	400	10
8	Mong Kok	500	40
10	Tsing Yi	400	15
12	Shatin	700	75
14	Tuen Mun	400	20
15	Tai Po	500	25
17	Sai Kung	400	10

18	Yuen Long	400	25
20	Tai O	600	25