COMP3211 Software Engineering

Monopoly

Software Requirements specification

Group 20

Tsang Cheuk Yu 24025237D

Wong Keng Wa 24018054D

Chan Ho Lok 24032915D

Content

[1.Preface 2](#_Toc183125267)

[Expected Readership: 2](#_Toc183125268)

[Version history: 2](#_Toc183125269)

[Introduction 3](#_Toc183125270)

[Purpose 3](#_Toc183125271)

[System function 3](#_Toc183125272)

[Objective 3](#_Toc183125273)

[Glossary 4](#_Toc183125274)

[User requirement definition 4](#_Toc183125275)

[User Functional Requirements 4](#_Toc183125276)

[User Non-Functional Requirements 5](#_Toc183125277)

[System architecture 7](#_Toc183125278)

[MVC 7](#_Toc183125279)

[Architectural Components 7](#_Toc183125280)

[System requirement specialization 8](#_Toc183125281)

[System Functional Requirements specialization(FRS) 8](#_Toc183125282)

[System Non-Functional Requirements Specification (NFRS) 9](#_Toc183125283)

# 1.Preface

## Expected Readership:

1. Game Developers: To implement the Monopoly game software based on the defined requirements.
2. Testers: To validate the system against the specified requirements and ensure functionality, usability, and correctness.
3. Stakeholders: To review and check the system’s objective and scoop to see if they fulfill the requirements.
4. User: To understand the features and the functions of the system and and how it fulfill their needs.

## Version history:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Rationale | Details of changes |
| V1.0 | 8/11/2024 | Initial draft | Document and graphs of basic system requirement based on user stories, rules and functions. |
| V1.1 | 14/11/2024 | Refinement | Finished coding of functional requirement of the game, a prototype has been done. |
| V1.2 | 17/11/2024 | Usability updates | Focus on the non-functional requirement, including consistency, responsiveness and compatibility, enhance usability. |
| V1.3 | 18/11/2024 | Bug fixes | Added saving and loading game state and fixes bug from the code. |
| V1.4 | 19/11/2024 | Testing | Testing different functions from the game to ensure no error happens. |

# Introduction

## Purpose

This need of this command-line based monopoly is an simulator of the traditional board game monopoly. Digitalized it helps simplified the gameplay, player can now customize features and rule. Therefore, player can interact and compete with other players in a virtual way with better user experience and enhance convenience.

## System function

System core functions are some gameplay mechanics, like rolling dice, player movement, player status and money transaction, autotomized it helps reduces complexity, error and chance of players cheating. In addition, the save game and load game function is also a key feature of the system which helps enhance user experience.

## Objective

Our system fits the objective of the COMP3211 course, in the other hand academic environment. It allows student to learn about OOP, handling about requirements of the users. In additional it promotes teamwork, enhance skill among software designing and development as well as time management among the large workloads.

# Glossary

Definition of the technical terms used in the document to avoid misunderstanding.

|  |  |
| --- | --- |
| Abbreviation | Meaning |
| V | Version |
| SRS | Special Requirement Specification |
| FR | Functional Requirement |
| NFR | Non-Functional Requirement |
| OOP | Object-Oriented Programming |
| I/O | Input/Output |
| SFR | System Functional Requirement |
| SNFR | System Non-Functional Requirement |

# User requirement definition

## User Functional Requirements

FR-1: The system shall allow players to start a new game using an existing game board, initializing all necessary game states such as player position, money, and property ownership.

 FR-2: The system shall enable players to input custom names or randomly generated name for each player at the start of a game.

FR-3: The system shall display the status of any player.

FR-4: The system shall provide a comprehensive view of the game state, showing the position, money, and property ownership of all players across the game board.

FR-5: The system shall determine and indicate the next player in the turn sequence, ensuring that turns proceed in the correct order from the first player to the last player.

FR-6: The system shall allow the player to save the current game into a file.

FR-7: The system shall the player to load a game from a file and continue the game.

FR-8: The system shall players to design their own game board by creating and organizing squares of seven types. Players can create a game board based on their hobbies.

FR-9: The system shall provide files where players can save their designed game boards.

FR-10: The system shall allow players to load a previously saved game from a file, restoring the game to its exact saved state to continue gameplay seamlessly.

## User Non-Functional Requirements

 NFR-1: The system shall process all player input, including dice rolls and game commands, within a minimum of milliseconds to ensure a responsive and smooth gaming experience.

 NFR-2: The system shall store saved game files in a human-readable and transferable format, such as JSON or XML

 NFR-3: The system shall support up to six players playing simultaneously without reducing performance.

 NFR-4: The system shall provide clear visual feedback for all player actions such as dice rolling, token movement, and block effects.

 NFR-5: The system shall be compatible with major operating systems, including Windows, macOS, and Linux, to ensure widespread use by users.

 NFR-6: The system shall ensure data integrity when saving and loading game state to prevent data damage or loss to maintain the reliability of game progress.

 NFR-7: The system shall be able to recover gracefully from errors, such as invalid input, without crashing.

 NFR-8: The system shall run with a minimal memory footprint to remain lightweight.

 NFR-9: The system shall have an intuitive and user-friendly interface that allows players to easily navigate game options, view status, and perform actions with minimal learning curve.

 NFR-10: The system shall be designed to be easily updated and maintained.

# System architecture

## MVC

The high level overview of our Monopoly Game system is the Model-View-Controller(MVC) pattern. It helps separate the presentation(view), interaction(controller) from the system data(model). It allows the data to change independently and support presentation of same data in different ways, if one changes is made, all other representation will be changed.

1. Model: It helps handle the logic and data representation of the Monopoly, Its functions include player information (name, balance, properties), gameplay mechanics (dice rolls, player movement, and transaction handling) and also some other gameboard elements. The model is reused for different features like customizable gameboard.
2. View: It manage the presentation aspect, interact with the player with the command-line interface, displaying all necessary information of the game with message as text. Functions include displaying player status and details ,I/O interaction(game event, buying property, end turn)and announcing game outcome. It is reused for enhancing the formatting.
3. Controller: It is the intermediary between the game and the player’s input. Functions includes process and validate player’s input, support game saving and loading, synchronize data between model and view as well as the gameboard customization and more. It is reused to enhance system functionalities and data flow.

## Architectural Components

Different Architectural Components are reused in our Monolopy system to support the functionalities.

1. Random Number Generator: determine the player movement by rolling dice by the random module
2. Game saves and Custom boards: data storage of game save and load as well as the customized gameboards are handles by JSON.
3. Error handling: handles invalid input and condition that is not expected with try-catch

# System requirement specialization

## System Functional Requirements specialization(FRS)

**FRS-1 Game Initialization:**

* The system shall allow players to start a new game using an existing game board, initializing all necessary game states, including player positions, money (HKD 1500 for each player), and property ownership (none initially).

**FRS-2 Player Customization:**

* The system shall enable players to input custom names or use randomly generated names for each player at the start of a game, ensuring clear differentiation among players.

**FRS-3 Player Status Display:**

* The system shall display the status of any specific player, including their current money, properties, and position on the game board.

**FRS-4 Game State Overview:**

* The system shall provide a comprehensive view of the game state, showing the position, money, and property ownership of all players across the game board.

**FRS-5 Turn Management:**

* The system shall determine and indicate the next player in the turn sequence, ensuring that turns proceed in the correct order from the first player to the last player, and cycling back to the first player after the last.

**FRS-6 Game Saving:**

* The system shall allow players to save the current game state into a file. The saved state shall include all player statuses, board configurations, and the current turn.

**FRS-7 Game Loading:**

* The system shall allow players to load a previously saved game from a file, restoring the game state, including all player statuses, board configurations, and the current turn, to enable seamless continuation of gameplay.

**FRS-8 Gameboard Customization:**

* The system shall allow players to design their own gameboards by creating and organizing squares of the seven predefined types (e.g., Property, Chance). This feature enables players to customize the gameboard according to their preferences or hobbies.

**FRS-9 Gameboard Saving:**

* The system shall provide functionality for players to save their designed gameboards into files, enabling future use or further customization.

**FRS-10 Gameboard Loading:**

* The system shall allow players to load a previously saved custom gameboard from a file, enabling them to use their customized board designs in a new game.

## System Non-Functional Requirements Specification (NFRS)

**NFRS-1 Responsiveness:**

* The system shall process all player input, including dice rolls and game commands, within milliseconds to ensure a responsive and smooth gaming experience.

**NFRS-2 File Storage Format:**

* The system shall store saved game files in a human-readable and transferable format, such as JSON or XML, to facilitate ease of use and compatibility across devices.

**NFRS-3 Performance Scalability:**

* The system shall support up to six players playing simultaneously without any degradation in performance.

**NFRS-4 Visual Feedback:**

* The system shall provide clear and immediate visual feedback for all player actions, such as dice rolling, token movement, and square effects, to enhance player engagement and clarity.

**NFRS-5 Platform Compatibility:**

* The system shall be compatible with major operating systems, including Windows, macOS, and Linux, ensuring accessibility for a wide user base.

**NFRS-6 Data Integrity:**

* The system shall ensure data integrity during save and load operations, preventing damage or loss of game progress to maintain reliability.

**NFRS-7 Error Handling:**

* The system shall recover gracefully from errors, such as invalid inputs, by providing meaningful error messages without crashing or losing progress.

**NFRS-8 Lightweight Operation:**

* The system shall run with a minimal memory footprint, making it lightweight and suitable for devices with limited resources.

**NFRS-9 Usability:**

* The system shall have an intuitive and user-friendly interface, allowing players to navigate game options, view statuses, and perform actions with minimal learning effort.

**NFRS-10 Maintainability and Extensibility:**

* The system shall be designed with modular and maintainable code to allow for easy updates, bug fixes, and the addition of new features or functionalities.