Software Requirements Specification

for

The Ultimate Maze Challenge

Version 1.0 approved

Prepared:

1. **Akshit Arora**
2. **Alejandro Olono**
3. **Zixuan Hu**

TCSS 360 B Team 1

11/10/2020

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Zixuan	11/12/20	Started SRS	1.0
Alejandro	12/17/20	Final implementation	1.1

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to provide a design of Maze game for kids. This document will provide the Client and the user with a basic understanding about the game's performance and its goals.

1.2 Document Conventions

When writing the SRS document for the Maze game, the following terminologies are: To make the document more effective and readable, I used the bold font style and font size, the title is bold and attractive colors are highlighted.

1.3 Intended Audience and Reading Suggestions

The document is intended for various kinds of readers, including the project manager, developers, testers, document writers, administrators of game organization, and others who are going to use this system.

For different readers, there are different reading suggestions. If you are the project manager or developers or testers, you'd better read throughout the document, that is to say, from the beginning to the end. If you are the game organizer, or you are just a player who is interested in this game, you only need to have a brief understanding of the system(what the game does, etc).

1.4 Project Scope

We are developing a maze game. The project will be developed using JAVA Swing.
We will program the character to transform through the maze in the game.
Our final product will deliver requirements, design documents and user documents, code and replacement files.

1.5 References

java swing documentation:
https://docs.oracle.com/javase/8/docs/technotes/guides/swing/index.html

sqlite-jdbc usage: https://github.com/xerial/sqlite-jdbc/blob/master/Usage.md

oracle jdbc documentation:https://docs.oracle.com/javase/8/docs/api/java/sql/package-summary.html

Stackoverflow: https://stackoverflow.com/

Geeksforgeeks Serialization Example: https://www.geeksforgeeks.org/serialization-in-java/

2. Overall Description

2.1 Product Perspective

The product to be made is a maze game where the player will attempt to reach the exit of the maze from the entrance where the player will start. Overall the game should be self-contained and dependency on external sources will be minimal.

2.2 Product Features

There will be a menu system that will allow players to save, load, and exit a game. In addition, the player will be able to seek help in an appropriately named tab named “help”. There will also be a helpful section on the screen that will provide detailed up-to-date information regarding the current room the player is in. Also, the player will be able to make a move depending on what moves are valid at the time. Finally, the player will have access to a section of the display where the question to be answered and the answer box will be housed.

2.3 User Classes and Characteristics

When it comes to who might play the game, overall it will likely be someone who can read and click and type on the keyboard. That is the vast majority of players who will likely be capable of playing the game and the parameters under which the game assumes are true of players. Though unlikely, there may also be players who are diabled and may not be able to read a question or make movements on screen.

2.4 Operating Environment

The program should be capable of running on the main OS’s and as such the operating system will likely not have an adverse effect on whether a player can run the game. The player should have access to the internet in order to download the game should the game be distributed as an executable online.

2.5 Design and Implementation Constraints

The game must use SQLite for the database and no other database software may be used. The game should be able to save the current state of the game. The game should have a maze size of at least four rooms by four rooms and there is no hard max size. Answers must be seeked by the game of the player for every new room that the player enters.

2.6 User Documentation

There will be an about page and a page describing how the game is played. Both pages will be accessible to the player from the menu system where selecting the “help” option should allow the player to view the desired help page.

2.7 Assumptions and Dependencies

It is assumed that the project will be able to be downloaded and the player will be able to do by method of running an executable. This can change however as time permitting this may or may not be implemented. The game doesn’t really depend all too much on external sources therefore feature implementation will depend solely on time available for development.

3. System Features

3.1 Maze Solver

3.1.1 Description and Priority

This feature would be responsible for the overall functionality of the game. It would provide the user with the graphical interface of the game and instruct the user how to play the game. This feature is of high priority. The benefit of this feature is that it would allow the user to input their responses and interact with the game. The benefit scale would be 9. The risk associated with this feature would be 4 out of 9 because it is responsible for the overall transition of the game, not the functionality of specific features within the game.

3.1.2 Stimulus/Response Sequences

***Stimulus**: The user requests to select the length of the maze game (From x mazes to up to y mazes (x, y is minimum and maximum length allowed, x, y TBD).*
***Response**: The system provides the user with the game size options to select from. The selected game is displayed.*
***Stimulus**: The user selects a door.*
***Response**: The system opens that door and provides the user with a question.*
***Stimulus**: The user answers the question.*
***Response**: The maze level proceeds if the question is answered correctly. The door is locked permanently if the answer is incorrect.*

3.1.3 Functional Requirements

FR-MS-1: Select the maze length requirement.

The system features an option to select the length of the maze game. The maze length (minimum and maximum TBD) should be strict. If the user tries to input invalid length, the user should be prompted to enter the correct length or exit the game. A default game length shall be decided as well (TBD).

FR-MS-2: Select the door and answer the question.
The user shall select a door from the options at a certain level. There would be a specific number of doors at each level (Number of doors TBD). The selected door should prompt the user to answer the question. If the answer is correct the user should be taken to the next level of the maze. If the answer is wrong, the door should be locked permanently. If the door doesn't contain any question, the user should be prompted to select another door.

FR-MS-3: Select the game view (current level, progress, complete game).
The system provides the user to visually see the current level, progress, and the complete view of the game. The user shall be able to select the view they want. If an invalid view is selected, the user shall be redirected to select the view again.

3.2 Random Question Generator

3.1.1 Description and Priority

The system shall generate the questions randomly from a list of questions depending on the level of the game. Whether a question shall contain the hint or not would be decided randomly as well. This feature is high priority because if the game has fixed questions, the return rate of the user would be less. This increases the cost of implementing and deploying this feature but would increase the functionality of the game and improve the overall user experience. We would rate the cost 6 out 9. The benefit of this feature is that it would increase the number of game plays for a user. We would rate the benefit 8 out of 9.

3.1.2 Stimulus/Response Sequences

***Stimulus**: The user selects the length of the game.*
***Response**: The system randomly generates the questions and their location in the game according to the maze depth. It also decides what questions shall contain hints.*
***Stimulus**: The user requests a hint.*
***Response**: The system provides a randomly generated hint from the list of hints.*

3.1.3 Functional Requirements

FR-RQG-1: Generate random questions for randomly selected doors at each level.
The system should decide what door has a question and what the question is. The difficulty of the game shall increase as the user progresses through the game.

FR-RQG-2: Generate random hints for the questions.
The system shall randomly decide what doors in the maze contain hints. The hints for the question in that door shall be randomly selected from the list of hints as well.

FR-RQG-3: Power door generator.
Few doors should contain powers such as skipping a level in the maze or ability to answer a question twice. These doors shall be decided randomly.

3.3 Game Saver and Game Loader

3.1.1 Description and Priority

The system shall allow the user to save their progress in a game under their credentials and reload it as per their convenience. This feature is of medium priority because implementing the game is the first priority. Adding functionality for saving and loading the gameplay at any time shall be implemented after the other two features. The benefit of this feature is user-convenience and multi-user functionality of the game. We would rate the benefit 6 out of 9. Some of the risks this feature might have are when two users try to save their progress under the same name, if they load someone else's game. We would rate the risk 4 out of 9.

3.1.2 Stimulus/Response Sequences

***Stimulus**: The user selects to save the progress in the game.*
***Response**: The system instructs the user to save the game and asks the user to input a name for the saved game.*
***Stimulus**: The user enters the name of the game.*
***Response**: The system saves the game if the name field is valid. The system instructs the user to enter the name again if invalid.*
***Stimulus**: The user selects to load a game.*
***Response**: The system displays all the saved games and asks the user to select the desired game.*
***Stimulus**: The user inputs the saved game to open,*
***Response**: If the input is valid, the system displays the game on the console or GUI (TBD). If the input is invalid, the game instructs the user to select the desired game again.*

3.1.3 Functional Requirements

FR-GSGL-1: Save the game.
The system shall provide the user with an option for saving the game. The user shall be instructed to input a valid name for the game. If the input name is invalid, the user shall be instructed to enter the name again.

FR-GSGL-2: Check whether the name is valid.
The system shall check whether the name for the saved game already exists or if the entered name is valid. It should display whether the name was accepted or what the errors in the name are.

FR-GSGL-3: Load game.
The system shall provide the user with an option to open the saved game. A list containing all the saved games should be displayed and the user shall input the game they want to load. If the selected game is invalid or broken, the issue shall be displayed and the user shall be instructed to select another game or to start a new game.

4. External Interface Requirements

4.1 User Interfaces

UI-1: The user's computer (MacOS/ Windows) display shall be used to to install the game and play it.
UI-2: The system shall permit the user to play the game either by using keyboard alone or a combination of keyboard and mouse commands.
UI-3: The system shall display any errors or warnings the game might incur on the user's computer display.
UI-3: The user shall be able to report any errors or suggestions via the game's discord channel. The discord server could be accessed via a phone, computer, or a tablet. Users could either use the discord's website or the application on either of the devices. Access to the internet is necessary for this requirement.

4.2 Hardware Interfaces

No hardware interfaces identified yet.

4.3 Software Interfaces

SI-1: The Maze Solver Game
SI-1.1: The Maze Solver shall let the user decide the length of the mazes. The inputted length shall be used to start the game.
S1-1.2: The Maze Solver shall randomly select what door contains what questions and whether the questions contain any hints or not. The system would do so by reading the questions and hints from a SQLite database which would contain a list of questions, their difficulty levels, and the hints for the questions. The user shall not be allowed to access or modify the database.
SI-1.3: Whenever the user tries to load a game and the saved game data is broken, the system shall display an error message to the console or GUI (TBD) and remove the erroneous game data.

4.4 Communications Interfaces

CI-1: The system shall display the game's social media links (Discord, Instagram) on the user's display.
CI-2: The social media links could be accessed using any device that has a stable internet connection (Mobile, Computer, Tablet). The user might require to create an account on these social media services to access the game's page.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Most operations should take less than a second. This includes; loading the game, bringing up help pages, loading and saving a game, and likely loading another room after answering a question correctly in a previous room.

5.2 Safety Requirements

Ensure that the project has gone through a complete unit test, and ensure that errors are correctly thrown away to ensure that there will be no unexpected uninterruptible loops in the project code that will affect the performance of the user's computer.

5.3 Security Requirements

The game does not have any networking function, and the user does not need to provide any personal information to play the game, so it will not cause any security issues

5.4 Software Quality Attributes

Availability: The usability of the program is very important to the user experience, so our goal is to achieve almost perfect Availability.
Correctness: The game does not allow players to pass through doors or rooms without answering the questions correctly.

Usability: The interface should be easy to play, without tutorials, and allow users to achieve their game goals without errors.
Testability: The code of the program should be modular and structured, and there should be independent unit tests for each module to verify its function.

6. Other Requirements

In order to not have conflicting types and overall a general working program, a javafx.base.jar is required to be in the build path so that the game may run. In case the javafx.base.jar file is not already in the build path, the file has been provided in the project so that in it may be manually added to the buildpath. To do so, right-click the overall project header from within package explorer and select “Build Path” -> “Configure Build Path” -> “Add jars” -> then select the javafx.base.jar, and select “Apply and Close”.