Software Requirements Specification

for

Trivia Maze

Version 1.0 approved

Prepared by Ji Lu, Jesse Flores, Xiying Long

TCSS 504 GROUP 3

March 17, 2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Ji Lu, Jesse Flores, Xiying Long	February 3, 2023	Initial Draft	1.0
Ji Lu, Jesse Flores, Xiying Long	March 17, 2023	Final Version	1.0

1. Introduction

1.1 Purpose

The Software Requirement Specification (SRS) for Trivia Maze is intended to serve as guideline and reference for the development team. It delineates product's functionality, features, external requirements and Non-functional requirements. Besides that, it also reflects design philosophy.

1.2 Project Scope

The project is aimed to delivering a graphical user interface (GUI) game named "Trivia Maze" to the player. It will help player navigate through the maze which consist of different size rooms from entrance to exit. To maximize entertainments of the game, one question at each door of the room is prepared for the player.

To win the game, player has to open all doors of rooms on their way from entrance to exit. Player either has to answer the question correctly to open the door of the room or choose using the "golden key". Player will only have one time opportunity to utilize the key per each game round. It is player's choice to decide when and where to use it. If all doors are locked and user runs out of the golden key, game will be terminated.

In order to achieve the goal, the project team has to implement but are not limited to developing a Unified Modeling language (UML) diagram, employing Model View Controller (MVC) design pattern, and incorporate SQLite database to store game information.

1.3 References

1. Guido van Rossum. Barry Warsaw. Nick Coghlan. PEP 8 – Style Guide for Python Code. https://peps.python.org/pep-0008/

2. Guido van Rossum. David Goodge. PEP 257 – Docstring Conventions. https://peps.python.org/pep-0008/

3. Paul, S. (2021, January 6). OpenTriviaQA

https://github.com/uberspot/OpenTriviaQA

2. Overall Description

2.1 Product Perspective and Product Features

As mentioned above, the maze is composed of a minimum 5-room by 5-room by default, however, it will not have an upper limitation. The difficulty level could be reset in "__init__" method under "maze.py" file. Each room will have different numbers of door and each door will be associated with one question. In order to successfully open the door, the player has to correctly answer the question or using golden key feature. If the player could not answer the question correctly and no more golden key can be utilized, door will be locked permanently. Questions will be presented in different ways that includes multiple choice questions, and True/False questions. In addition, questions will cover few different categories.

2.2 User Classes and Characteristics

Room	The room class will include the following attributes: exit, row, col, north, south, west, and east. When the player enters a different room and answers different questions, the room will update its attributes accordingly. Implementation will include a golden key function that allows access to next room despite failing to answer a question.
Maze	The maze class will hold a 2-dimensional array which used to represent the maze in the current state for the game, and each cell in the array will hold a room object.
Player	The Player will keep all the properties relevant to the players progression, such as how many golden keys they have.
Question	The Question will connect to existing SQLite database and retrieve all questions and associated answers.
QuestionAnswer Database	The QuestionAnswerDatabase will set up SQLite database which holds all questions and answers.
Controller	Controller will be used to control the whole game's logic.

ShapeTests	The ShapeTests class will be served as unit test to ensure all functions within Player's class functioning correctly.

2.3 Operating Environment

OE-1: The game shall operate with python version 3.x to compile the python script and be compatible with macOS and Windows operating system.

OE-2: The trivia maze game shall have a NoSQL server running and able to make a query on the database.

2.4 Design and Implementation Constraints

CO-1: The code shall conform to PEP 8 style guidelines for python code. Additionally, the code should be separated on its own .py file as necessary and should reflect Object Oriented Design Patterns.

CO-2: The maze game should incorporate the NoSQL database standard.

CO-3: A python GUI to create user interfaces using native elements for this application.

CO-4: All scripts shall be written in python.

2.5 User Documentation

UD-1: When the game is initiated, an introduction section appears on bottom of the window followed by: New Game, Load Game, Instructions and Exit.

UD-2: During the game, a cluster of buttons will be provided on the upper left corner to give player's choice to choose. For instance, player could re-start the game by clicking New Game button or save the current game by clicking Save Game.

UD-3: A game instruction is holding all the game rules.

2.6 Assumptions and Dependencies

DE-1: The game is depending on how the database is formatted, and how we query for the data.

3. System Features

3.1 Opening the door for access

3.1.1 Description and Priority

A player starts at a random location in the maze. The initial state should be 4 doors in a room (general case), and each door will be displayed. The player can choose any one of 4 doors to precede the game, question will be pop up once door is selected. When the question is answered correctly, the door will be shown in an open state. If the answer is wrong, the door will displayed in a closed state. Priority = High.

3.1.2 Stimulus/Response Sequences

Stimulus: The player starts at the entrance and selects a door to access.

Response: A query will be fetched from database, and one question will be displayed.

Stimulus: The player selects an answer to the question.

Response: The door will open if the question is correct. The player is able to access to next room. The door will close if the answer is wrong. The player can choose other available options.

3.1.3 Functional Requirements

North_Door: The Door attributes will be Exit, Open, and Close three states. GUI will display the appropriate door visual status based on the door attribute value.

South_Door: The Door attributes will be Exist, Open, and Close three states. GUI will display the appropriate door visual status based on the door attribute value.

West_Door: The Door attributes will be Exist, Open, and Close three states. GUI will display the appropriate door visual status based on the door attribute value.

Eest_Door: The Door attributes will be Exist, Open, and Close three states. GUI will display the appropriate door visual status based on the door attribute value.

Row: It will contain the current player row number

Col: It will contain the current player col number

Visited: Once the player leaves the room, it will be marked as visited

3.2 Golden Key

3.2.1 Description and Priority

When player answers all questions wrong and is stuck in the room, they could choose to use the golden key to access to the next room. Or when the player is approaching the exit. They can decide to use the golden key directly opening the door to access to the exit. Priority = median

3.2.2 Stimulus/Response Sequences

Stimulus: The player can decide to use a golden key.

Response: When a golden key is available, it opens a door and updates the availability of the key for future uses.

3.1.3 Functional Requirements

Golden_key_is_used: A Boolean value is used to show whether the golden key is used or not.

4. External Interface Requirements

4.1 User Interfaces

UI-1. The TriviaMaze GUI shall have one menu system. The menu system will contain at least two tabs which are File and Help. Under File tab, it shall have choices which include Start New Game, Save Current Game, Load Last Game and Exit Game. Under Help tab, it shall display About and Game Instruction choice.

UI-2. The GUI shall show the player the current room they are located.

UI-3. The GUI shall include a question section which displays the question provided to the player.

UI-4. The GUI shall display feedback if player's answer is correct and automatically move the player to next room.

UI-5. The GUI shall display feedback and correct answer when the player's answer is incorrect.

UI-6. The GUI shall prompt the player with a message "You have 1 golden key available! Pick a door to unlock." when key is available to use and all doors are closed.

UI-7. The GUI shall prompt a message box asking the player wants to re-play when game is over if all doors are locked and key is not available to use.

4.2 Hardware Interfaces

1. Portions of game data shall be stored in a SQLite database.

2. The TriviaMaze GUI shall be able to save the current game and load the game saved from last time.

4.3 Software Interfaces

SI-1. The GUI Interface shall pass all necessary information to Class Maze to generate appropriate size of maze.

SI-2. Class TriviaMazeGUI will be called if the user wants to initiate game or get game instruction. Player's answer shall also be checked here.

SI-3. Class Player shall get and set information such as points the user has and transmit it back to TriviaMaze GUI Interface to show the player wins or loose.

SI-4. Class Questions shall be used to connect to SQLite database and to retrieve all questions and correct answers.

SI-5. Information for size of maze is set up under class Maze. It will be passed to Class Room and it shall generate all necessary rooms and set doors in each room.

4.4 Communications Interfaces

CI-1. The TriviaMaze GUI shall generate message boxes to the user displaying the information the player wants to know, such as about, and game instruction.

CI-2. The TriviaMaze GUI shall send feedback confirmation message regarding the choice made by the player.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

1. The software shall display updated information, game status, and available buttons based on the player's interactions throughout the progression of the game; each new render show display within 20 seconds.

2. Questions shall be loaded from SQLite database and shall take no longer than 20 seconds to render on screen.

5.2 Safety Requirements

For project wise purpose, safety requirements does not apply to this project.

5.3 Security Requirements

1. The game data is stored in a database and is not directly accessible for edits via the GUI.

5.4 Software Quality Attributes

1. The program shall allow the player to exit the game whenever the user decides to do so.
2. The program shall allow the player to exit the game with saving their current status.
3. The program shall allow the player to load the game from last time they saved.
4. The program shall allow the player to restart a game at any time.

Appendix A: Glossary

GUI	Graphical User Interface, a user interface that allows interaction between the user and software.
User or Player	Any person or group utilizing the software.
UML	Unified Modeling Language (UML), a blueprint for visualizing the software design.

Appendix B: Analysis Models

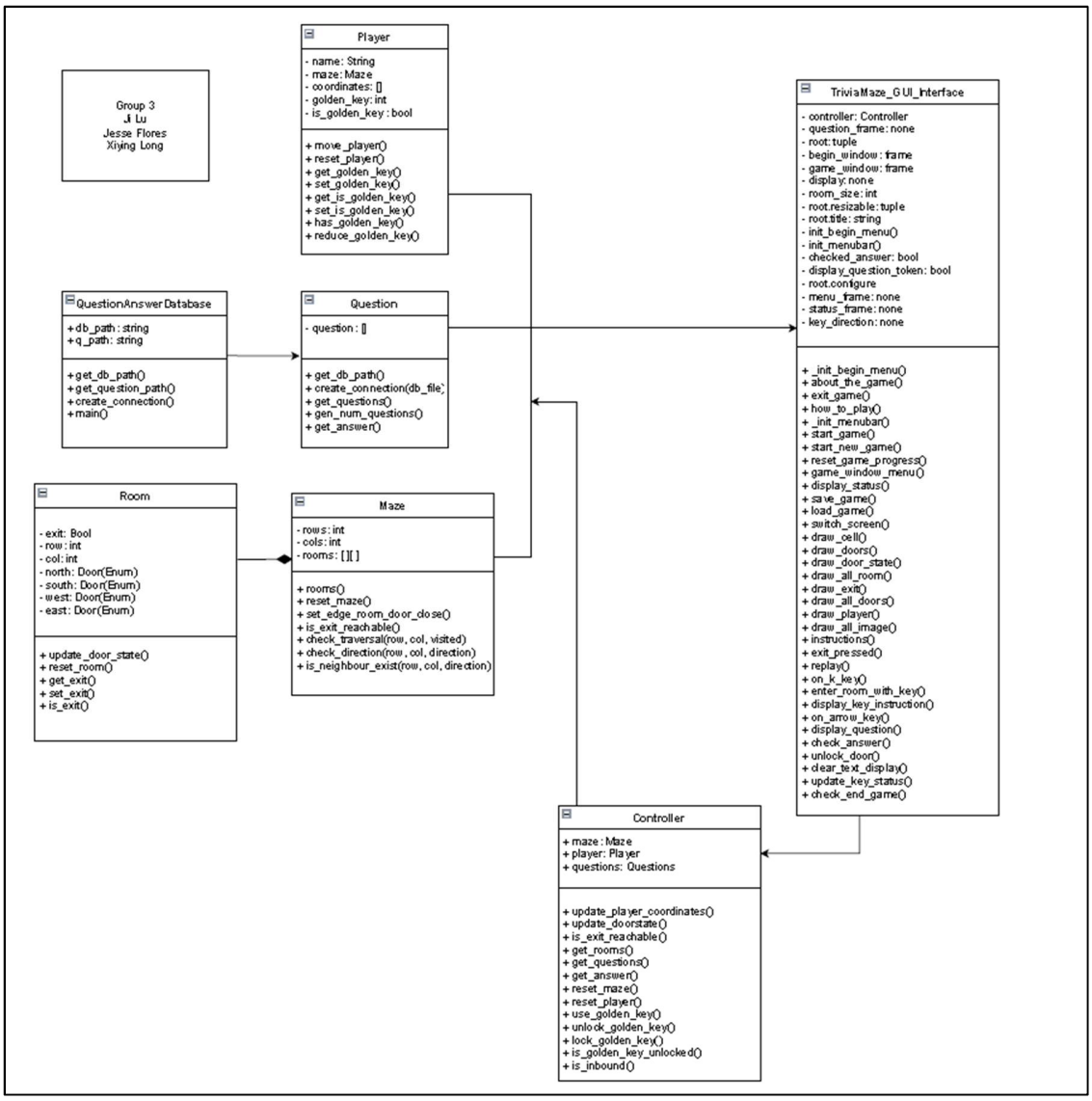

Figure 1UML diagram for Trivia Maze