Software Requirements Specification

for

Trivia Maze

Version 1.0 approved

Prepared by Ji Lu, Jesse Flores, Xiying Long

TCSS 504 GROUP 3

February 3, 2023

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Project Scope 1

1.3 References 1

2. Overall Description 1

2.1 Product Perspective and Product Features 1

2.2 User Classes and Characteristics 1

2.3 Operating Environment 2

2.4 Design and Implementation Constraints 2

2.5 User Documentation 2

2.6 Assumptions and Dependencies 3

3. System Features 3

3.1 Opening the door for access 3

3.2 Golden Key 4

4. External Interface Requirements 4

4.1 User Interfaces 4

4.2 Hardware Interfaces 4

4.3 Software Interfaces 5

4.4 Communications Interfaces 5

5. Other Nonfunctional Requirements 5

5.1 Performance Requirements 5

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

Appendix A: Glossary 6

Appendix B: Analysis Models 6

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Ji Lu, Jesse Flores, Xiying Long | February 3, 2023 | Initial Draft | 1.0 |
|  |  |  |  |

# Introduction

## 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.

## Project Scope

The project is aimed to delivering a graphical user interface game named Trivia Maze to the user. It will help user navigate through the maze which consist of different size rooms from entrance to exit. To maximize entertainments of the game, one question at each room is prepared for the user. User has to answer the question correctly to pass the room and eventually win the game; otherwise, 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.

## References

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

# Overall Description

## Product Perspective and Product Features

As mentioned above, the maze is composed of different size of rooms at a minimum 4-room by 4-room, however, it will not have an upper limitation. The difficulty level is depending on user’s choice. Each room will have one door and one question. In order to successfully pass the room, the user has to correctly answer the question. If the user could not answer the question correctly, door will be locked permanently. Questions will be presented in different ways that includes multiple choice questions, True/False questions, and even fill the blank questions. In addition, questions are well developed by project team and it will cover different categories.

## User Classes and Characteristics

|  |  |
| --- | --- |
| Room | The room class will include the attributes: exit, entrance, row, col, state of doors. 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, and each cell in the array will hold a room object. The 2-dimensional array will represent all rooms in the current state for the game. We also added a maze difficulty attribute. |

|  |  |
| --- | --- |
| Player | The Player will keep all the properties relevant to the players progression, such as how many golden keys they have. |
| Question\_Answer\_DB | The Question\_Answer\_DB will connect to the database we set up. Provides a way to retrieve information from the database via methods. |
| TriviaMaze\_GUI\_Interface | The TriviaMaze\_GUI\_Interface uses a python GUI native element to create the user interface and interaction with the Room class and Player class. It displays all available rooms and the player's current location. Another important portion is it will listen to the events that the player triggers by using event\_check(). The states will update when changes in the Room's attribute and the Player's attribute occur. |

|  |  |
| --- | --- |
| TriviaMaze | The TriviaMaze class will include the introduction section, method to initiate the game, and method to restart the game. |

## 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.

## 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.

## User Documentation

UD-1: When the game is initiated, an introduction section appears on the window followed by: the game rules, users win/lose stats, entrance keys, and game questions.

UD-2: During the game, an info button is provided to give additional information regarding the game such as a game introduction. They can look back for the information if players forget about the rules.

UD-3: A user manual is to include all the game rules and software setup for starting the game.

## Assumptions and Dependencies

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

# System Features

## 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 one door to precede, one question will pop up. When the question is answered correctly, the door will be in the open state. If the answer to the question is wrong, the door will display a close 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 make to the database, and a 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 will have access to another room. The door will close if the question 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 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 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 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 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

## Golden Key

3.2.1 Description and Priority

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

# External Interface Requirements

## User Interfaces

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.
2. The GUI shall have a legend section to display what all key symbols stand for to the user.
3. The GUI shall include a map section which permit the user navigating through from entrance to exit.
4. The GUI shall also have a section which display the information of the current room.
5. The GUI shall include a question section which displays the question provided to the user.
6. The GUI shall display feedback when the user response is correct and shall automatically move the user to that room.
7. The GUI shall display feedback when the user response is incorrect.
8. The GUI shall prompt the user with a message “You have a golden key available, do you want to use it?” when there are keys available to use and all doors a closed.
9. The GUI shall provide feedback that the game is over if all doors are closed and there are no available keys to use.

## 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 save from last time.

## Software Interfaces

1. The TriviaMaze\_GUI\_Interface shall pass all necessary information to Class Maze to generate appropriate size of maze.
2. Class TriviaMaze will be called if the user wants to initiate game or get game instruction.
3. Class Player shall get and set information such as points the user has and transmit it back to TriviaMaze\_GUI\_Interface to display the user wins or not.
4. Class Question shall be used to collect question and correct answer associated with it from SQLite database. User’s answer shall also be checked here.
5. Once the size of maze is determined by the user, the information will be passed to Class Room and it shall generate all necessary rooms and set doors in each room.

## Communications Interfaces

1. The TriviaMaze GUI shall display a dialog box to the user to display the information the user wants to know.
2. The TriviaMaze GUI shall send feedback confirmation message regarding the choice made by the user.

# Other Nonfunctional Requirements

## Performance Requirements

1. The software shall display updated information, game status, and available buttons based on the user’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.

## Safety Requirements

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

## Security Requirements

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

## Software Quality Attributes

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

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

Diagram

Description automatically generated

**Diagram 1**

*UML diagram for Trivia Maze*