Software Requirements Specification

for

Winter Olympics Trivia Maze

Version 1.0 approved

Prepared by “The Four Musketeers”: Shine May Hung, Yan Wang, Manuel Duarte, Michael Weinberg

UW Tacoma Winter 2022 TCSS 504 Group Project

01/28/2022

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Project Scope 1

1.5 References 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Features 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

3. System Features 3

3.1 System Feature 1 3

3.2 System Feature 2 (and so on) 4

4. External Interface Requirements 4

4.1 User Interfaces 4

4.2 Hardware Interfaces 4

4.3 Software Interfaces 4

4.4 Communications Interfaces 4

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

6. Other Requirements 5

Appendix A: Glossary 5

Appendix B: Analysis Models 6

Appendix C: Issues List 6

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

This documentation identifies the software requirements for our Winter Olympics Trivia Maze game version 1.0. This game is run with Python 3.9, uses a SQLite version relational database, and is developed in the PyCharm environment. The program uses Model View Controller architecture. Python code runs on the back end and Angular/JavaScript is used on the front end. SQLite stores the questions and answers for the trivia part of the game.

## Document Conventions

Do fonts or highlighting in this document have special significance? We will determine this as this document evolves.

## Intended Audience and Reading Suggestions

The intended audience for this game are the instructor for TCSS 504 Winter 2022, Tom Capaul, fellow classmates, and other interested parties. The rest of the SRS contains an introduction, overall product description, system features, external interface requirements, other non-functional requirements, and appendices. It is organized categorically and the suggested sequence for reading the document is to begin with the overview sections and then proceed through the sections that are most pertinent to the reader.

## Project Scope

The Winter Olympics Trivia Maze game seeks to promote knowledge about athletes and events of the historical Winter Olympics. Its goals are to entertain and inform users of all ages and will benefit and enrich the lives of those who play. No corporate goals or business strategies thus far.

## References

References to be identified and spelled out as we use them.

# Overall Description

## Product Perspective

The Winter Olympics Trivia Game will consist of two parts: a database to store questions and answers, and a GUI interface for the user to interact with the game. The GUI will make the game fun and easy to use. In addition, it offers a number of options such save and restart game as well as other helpful elements to the user.

## Product Features

The Winter Olympics Trivia Game helps sports enthusiasts learn and engage with fun facts from the historical Winter Olympic games. You can learn all about your favorite winter sports, athletes, and new performance records as you play this game. There is a timer feature to challenge your knowledge.

## User Classes and Characteristics

The Winter Olympics Trivia Game (WOTG), will allow the user(s) to play a fun trivia-style game and at the same time increase their current knowledge about the history of the Winter Olympics. It offers players of all ages an opportunity to increase their knowledge about one of the most popular sports events of our planet. WOTG will offer the player both an offline mode and online mode playing experience.

## Operating Environment

OE-1: The Olympic Trivia Game shall run on the following Operating Systems: Windows 8, Windows 10, Mac OS 10.14 to Mac OS 12, and Linux Ubuntu.

OE-2: All code shall comply with up-to-date PEP specifications for best practices for better performance.

OE-3: ALL programming shall be written using Python 3.6 or greater.

OE-4: The UI shall be developed using the latest versions of Tinker.

OE-5: Sound shall be developed using PyGame and its libraries

## Design and Implementation Constraints

OTG initial design runs on Tinker of the most popular IU available to Python today. Tinker allows us to create user friendly interfaces to make game playing easier play for all users, but sadly no support will be included at this time. To play with the most current trivia information available future option to update the current data by updating online might be available in next 2023 version.

## User Documentation

UD-1: OTG will provide a well-documented tutorial available on the (UI) user interface.

UD-2: A list of minimum system requirements will be available at initial runtime.

UD-3: Instructions will include to help navigate OTG and to make sure you get the most out of our game.

## Assumptions and Dependencies

One assumption is that this product will always be used on desktops and laptops with enough memory and CPU power to run the application. Upgrades and increase in RAM will improve the player (s) experience. Remember to play hard and stay in school.

# System Features

## Show current Map and Player

3.1.1 **Description and Priority**

Draw a Trivia Maze with a lot of rooms for the players to explore. A player may move in the map if it has not yet reached the edge. Priority = High.

3.1.2 **Stimulus/Response Sequences**

Stimulus: Player requests to move right.

Response: System queries Player for location and the validity of room on the right.

Stimulus: Player requests to move down.

Response: System queries Player for location and the validity of room on the bottom.

Stimulus: Player requests to move left.

Response: System queries Player for location and the validity of room on the left.

Stimulus: Player requests to move up.

Response: System queries Player for location and the validity of room on the top.

3.1.3 **Functional Requirements**

|  |
| --- |
| Player.Move: The system shall let a Player move to the Room nearby.  Player.Move.Left: Move Player to the Room at Left.  Player.Move.Right: Move Player to the Room at Right.  Player.Move.Up: Move Player to the Room at top.  Player.Move.Down: Move Player to the Room at bottom. |
| Map.Player.Location: Map should show where the Player is.  Map.Room: Map should show all the Rooms.  Map.Room.Block: If the Room is blocked, the system shall give the Player a message to show the Room has already been visited.  Map.Room.Empty: If the Room is empty, the system shall let the Player move to that Room after Player passes the Challenge.  Map.GameOver: No Room is available around the Player.  Map.Player.Win: Player reaches the End point. |

## Show challenging Questions

3.1.1 **Description and Priority**

To go to another Room, the Player should answer a question. If the Player select a wrong answer, then the Room would be blocked. The Player should choose another Room to go. If the answer is right, the Player will move to that Room. Priority = High.

3.1.2 **Stimulus/Response Sequences**

Stimulus: Player requests to enter a Room.

Response: System queries Database to show a Question.

Stimulus: Player select wrong answer.

Response: System deny Player’s movement to the Room.

Stimulus: Player select right answer.

Response: System move the Player into the Room.

Stimulus: Player select help.

Response: System show clue for the Question.

3.1.3 **Functional Requirements**

|  |
| --- |
| Question.Right: The system shall let Player move to new Room.  Question.Wrong: The system shall mark the Room invalid, and Player stays in the original Room.  Question.Help: The system shall prompt the Player with the answer to the Question. |
|  |

## Show updated Map and Player

3.1.1 **Description and Priority**

Update the location of the Player and Room availability in the Trivia Maze. Priority = High.

3.1.2 **Stimulus/Response Sequences**

Stimulus: Player move to a new Room.

Response: System queries Player for new location and update the Map.

Stimulus: Room turned invalid after choosing wrong answer.

Response: Update the Room to let Player know it is blocked.

3.1.3 **Functional Requirements**

|  |
| --- |
| Map.Player.Location.Update: Map should show where the Player is.  Map.Room.Update: Map should show updated Rooms. |

# External Interface Requirements

## User Interfaces

To be determined.

## Hardware Interfaces

Application will be displayed via desktop or mobile web browser.

## Software Interfaces

Proposed:

SI-1: Trivia Maze Service

SI-1.1: The graphical user interface will transmit input data from the user to the trivia maze service.

SI-1.2: The trivia maze service will communicate with a relational database to fetch data requested by the user without the user directly interacting with the database.

SI-2: Relational Database

SI-2.1: The relational database will perform CRUD operations on data pertinent to the application.

## Communications Interfaces

None so far.

# Other Nonfunctional Requirements

## Performance Requirements

To be determined.

## Safety Requirements

To be determined.

## Security Requirements

To be determined.

## Software Quality Attributes

To be determined.

# Other Requirements

Appendix A: Glossary

To be determined.

Appendix B: Analysis Models

UML class diagram to go here.

Appendix C: Issues List

To be determined.