TRIVIA MAZE: THE GAME

# Software Requirement Specifications

Version 0.1

Prepared by:

Nate Hiblar

Quin Tiller

Wyran Parr

Date Created:

5.15.2020

Table of Contents

[**SOFTWARE REQUIREMENT SPECIFICATIONS 0**](https://d.docs.live.net/6b348535d876d8cd/Documents/EWU/Winter%202017/CSCD%20488/SRS/SRS-Rough-Draft%20v3.docx#_Toc479931547)

1. [INTRODUCTION 3](#_bookmark0)
   1. [PURPOSE 3](#_bookmark1)
   2. [PROJECT SCOPE 3](#_bookmark4)
2. [OVERALL DESCRIPTION 4](#_bookmark6)
   1. [PRODUCT PERSPECTIVE 4](#_bookmark7)
   2. User Classes
   3. [OPERATING ENVIRONMENT 4](#_bookmark9)
   4. [DESIGN AND IMPLEMENTATION CONSTRAINTS 4](#_bookmark10)
   5. [ASSUMPTIONS AND DEPENDENCIES](#_bookmark11) 4
3. [SYSTEM FEATURES 6](#_bookmark12)
   1. [FEATURE 1 6](#_bookmark14)
      1. [DESCRIPTION AND PRIORITY 6](#_bookmark15)
      2. [STIMULUS AND RESPONSE 6](#_bookmark16)
      3. [FUNCTIONAL REQUIRMENTS 6](#_bookmark17)
   2. [FEATURE 2 6](#_bookmark14)
      1. [DESCRIPTION AND PRIORITY 6](#_bookmark15)
      2. [STIMULUS AND RESPONSE 6](#_bookmark16)
      3. [FUNCTIONAL REQUIRMENTS 6](#_bookmark17)
   3. FEATURE 3
      1. [DESCRIPTION AND PRIORITY 6](#_bookmark15)
      2. [STIMULUS AND RESPONSE 6](#_bookmark16)
      3. [FUNCTIONAL REQUIRMENTS 6](#_bookmark17)
4. [EXTERNAL INTERFACE REQUIRMENTS 12](#_bookmark33)
   1. [EXAM STATES 12](#_bookmark34)

|  |  |  |
| --- | --- | --- |
| [5.1.1](#_bookmark35) | [HIDDEN](#_bookmark35) | [12](#_bookmark35) |

1. [**EXTERNAL INTERFACE REQUIREMENTS 13**](#_bookmark45)
   1. [**HARDWARE INTERFACES 13**](#_bookmark46)
   2. [**SOFTWARE INTERFACES 13**](#_bookmark47)
   3. [**USER INTERFACES 13**](#_bookmark48)
   4. [**COMMUNICATION INTERFACES 13**](#_bookmark49)
2. [**NON-FUNCTIONAL REQUIREMENTS 14**](#_bookmark50)
   1. [**NONFUNCTIONAL REQUIREMENTS 14**](#_bookmark51)

# Introduction

## Purpose

The purpose of this SRS document is to detail all known requirements and relevant information for our team to develop a trivia-based maze game. This information will include items such as game features, interfaces, and functions.

## Project Scope

The scope of this project is to design and create a desktop-based trivia maze game which will be comprised of the game itself as well as administrator tools for managing the game’s question database. The software will include an installer program and will be simple to both install and use by any players or game administrators. A player should be able to load an existing game, start a new game, or quit at any time. While playing the maze, a player should be able to traverse the maze from start to finish, and answer questions while traversing. When the player completes the maze, or loses a game, they should also be able to exit the program or restart the game. An administrator should be able to add, delete or modify questions inside of the database that will be used by the game to present questions to the player.

# Overall Description

## Product Perspective

The Trivia Maze game is meant to provide players with a navigable maze game which requires answering trivia questions correctly to successfully traverse. It will provide this functionality through an easily used GUI interface. Additionally, the game will provide basic functions accessible for administration purposes to permit adding, deleting, or altering the questions which are utilized by the game.

* 1. User Classes

Player: A player will start or load an instance of the trivia maze game, traverse the maze in an instance of a game and answer trivia questions while traversing the maze.

Administrator: An administrator will add, modify, or delete questions and answers that exist inside of a database file which provides the questions the player gets prompted with.

## Operating Environment

|  |  |
| --- | --- |
| OE-1:  OE-2: | The game will be compatible with the Windows 10 operating system.  The system will require the Java Runtime Environment. |

## Design and Implementation Constraints

|  |  |
| --- | --- |
| CO-1:  CO-2: | The game shall use ­­­­­­­­SQLite JDBC as its database.  Code written for the game shall be in Java. |

## Assumptions and Dependencies

|  |  |
| --- | --- |
| AS-1: | It can be assumed that the database will contain enough questions to enable the game to function. |
| DE-1: | The game is dependent on a .db database file to be present. |

# System Features

## GUI

3.1.1 Description and Priority

Features will include, saving the game, loading the game, being able to navigate the maze using mouse, will consist of short answer and multiple choice questions, there will be a GUI for all of this and it will consist of 4 multiple choice buttons that you can click to submit your answers to get through the maze. Furthermore, locking and unlocking doors for each room based on answer choice. I believe all these are high priority since this game would not be much of a game without these functionalities.

3.1.2 Stimulus/Response Sequences

Using mouse to click arrows to move from room to room or the maze to move from room to room.

Receiving and answering the question to continue with the game.

Not answering the question correctly will Block your path until eventually there will be no more paths to take and at this point you have lost.

Answering correctly will allow the locked rooms to be unlocked.

Saving the game so you can continue later but also keep your work.

Loading the game so you can possibly have multiple games you want to finish.

Restarting the game if you feel like you have already made to many mistakes.

Quit the game.

3.1.3 Functional Requirements

REQ-1: There will be edge cases that only allow mouse use for navigating through the maze and selecting answers.

REQ-2: Depending on the questions we ask, there should be edge cases for how many and what kind of characters you may use on the short answer questions.

REQ-3: In order for the game to be played later we must implement save game and load game, which will most likely be a drop-down menu at the top.

REQ-4: If we want the player to know where he is in the maze, we must create a game piece and create a method that will continually check the players current location until the game is won.

REQ-5: We also want the player to know when something is answered wrong or right and notify the player as such. If an answer is wrong the room will clearly display a blocked path and if the answered question is correct it will clearly show the game piece moving forward.

REQ-6: Allowing a restart, there must be a way for the player to click the menu and have the option to refresh the game.

## Database

3.1.1 Description and Priority

The database shall hold questions and answers of three different question formats that will be utilized by the system as prompts for the player during use of the program. Additionally, the database will permit an administrator to add, delete or modify questions and answers in the database. The database is a high priority of the feature of the system.

3.1.2 Stimulus/Response Sequences

Stimulus: System queries for a question of a given type.

Response: A question of the type requested is returned along with its corresponding answer.

Stimulus: Administrator requests deletion of a particular question/answer.

Response: Database deletes the table entries corresponding to the specified question/answer.

Stimulus: Administrator adds a question/answer to the database.

Response: Database updates the corresponding table with the new question/answer.

Stimulus: Administrator updates a question/answer.

Response: Database updates the question/answer in its corrisponding table.

3.1.3 Functional Requirements

Database.query: The database takes in a query for a question of a specific type.

Database.question: The database returns a question from the database of requested type.

Database.answer: The database returns the answer to a corresponding question.

Database.addMulti: The database adds a new multiple-choice table entry with the user input specified for both the question and answers.

Database.addTF: The database adds a new true-false table entry with the user input specified for both the question and answer.

Database.addShort: The database adds a new short answer table entry with the user input specified for both the question and answer.

Database.update: The database updates a specified question and/or answer that exists inside one of its table given the user input.

Database.delete: The database deletes a specified question and answer given the user input.

# External Interface Requirements

## Hardware Interfaces

The game will run on any computer running windows 10.

## Software Interfaces

SI-1: The trivia maze will query the database for a given question and answer type.

SI-2: The database will send the resulting question and answer to the trivia maze as return strings.

SI-3: When a question has been used, the database shall mark that the question should not be used as valid return for a maze query.

SI-4: The trivia maze shall refresh the database upon restart of the game.

SI-5: Upon notification from the trivia maze of a refresh, the database shall unmark any questions in the database to make them available for query again.

## User Interfaces

UI-1: The system will use a GUI for the player to interact with the software.

UI-2: The system will use a text based interface in order to interact with the database.

## Communication Interfaces

\_\_\_\_\_\_\_\_\_\_\_\_\_

.

# Non-Functional Requirements

## Other Requirements

## Performance Requirements

PE-1: The system shall queue questions for the player from the database with minimal to no delay.

PE-2: The system shall update the maze within 2 seconds of the player submitting an answer to a question.

PE-3: The database shall update within 2 seconds of the game administrator submitting a change (alteration, deletion, addition).

## Safety Requirements

There are currently no safety requirements for use of the software.

## Security Requirements

SE-1: The database will confirm that the administrator wants to make a change to the database before submitting the change to the database.

## Software Quality Attributes

Availibility-1: The software shall hold the last previously saved game until such time as the user saves the game again (whether from a re-load or new game).

Availibility-2: The database will hold all questions and answers that have been added into the database file until said file is deleted or otherwise altered by the game administrator.

Robustness-1: The software shall handle any misshapen input by prompting the user to re-enter their input.

# Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>