Software Requirements Specification

for

Harry Potter Trivia Maze

Version 1.2 approved

Prepared by Kayla Foremski, Calum Mackenzie, and Roon Le

Team HardMode

December 9, 2019

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
First Draft	11/6/19	N/A	1.0
Second Draft	11/24/19	UML Change	1.1
Third Draft	12/8/19	UML and Program Changes	1.2

1. Introduction

1.1 Purpose

This SRS documents all the requirements and necessary information for what Team HardMode will produce as the Trivia Maze final project. It will explain in detail the features and overall workings of the system.

1.2 Document Conventions

The requirements listed in this document are intended to be fulfilled and in production by the final project due date of December 10, 2019.

1.3 Intended Audience and Reading Suggestions

This documentation's intended audience is Tom Capaul, Team HardMode, and the class of CSCD 350.

1.4 Project Scope

The scope of this project pertains to CSCD 350's final project specifications and requirements. The software will be simple for users to play. It will allow users to interact with a Trivia Maze where each room will ask a trivia-style question. If a user successfully answers, the room will unlock allowing the user to progress through the maze. If the user answers incorrectly, the room will permanently lock, requiring an alternate maze route. The end goal for the user to successfully exit the maze.

2. Overall Description

2.1 Product Perspective

The Trivia Maze project will provide users an interactive, fun game to test their trivia knowledge on Harry Potter.

2.2 Product Features

Main features include a maze with locking rooms and trivia questions that will be in the form of multiple choice, short answer, and true/false. The user will also be able to save and load their game state in addition to adding questions to the question database.

2.3 User Classes and Characteristics

User Will interact with the maze when they encounter a room. Will be required to answer a trivia question.

2.4 Operating Environment

OE – 1: The system will have compatibility with C# .NET framework using Visual Studio 1.38.1.

2.5 Design and Implementation Constraints

CO – 1: The system will use System.Data.SQLite 1.0.112.0 as its database.

CO – 2: The program will be written in C# version 6.

3. System Features

3.1 Board

3.1.1 Description and Priority

High priority feature consisting of a 4x4 2D array of rooms. Each board will also contain a player.

3.1.2 Stimulus/Response Sequences

Users will enter the 2D array and will have to answer questions in each room to traverse through it in a maze-like fashion.

3.1.3 Functional Requirements

REQ-1: Maze must be 4x4 at minimum.
REQ-2: Must have an entrance room (top left) and an exit room (bottom right by default).
REQ-3: Must allow users to traverse through, using arrow keys on keyboard.

3.2 Player

3.2.1 Description and Priority

Player represents a user and their associated Hogwarts House which will determine the color scheme of the program.

3.2.2 Stimulus/Response Sequences

Users must select their associated Hogwarts House at the beginning of the program. This will be assigned to the Player.

3.2.3 Functional Requirements

REQ-1: Must have an associated Hogwarts House.

3.3 Room

3.3.1 Description and Priority

High priority feature consisting of a locked or unlocked and attempted and un-attempted state. Each room will have a number to associated it with a specific question.

3.3.2 Stimulus/Response Sequences

Users will have to answer a question when encountering a locked and un-attempted room. If the question is answered successfully, the room will unlock and become attempted, if it is answered incorrectly, the room will remain locked and attempted for the remainder of the game.

3.3.3 Functional Requirements

REQ-1: Entrance room will be unlocked and attempted (top left room).
REQ-2: All other rooms will be locked and un-attempted by default.
REQ-3: Every time a user tries to enter a new room (room state locked and un-attempted), a question will be triggered.
REQ-4: If a user has successfully entered a room before (room state is unlocked and attempted), the room will allow users to enter freely.

3.4 Question

3.4.1 Description and Priority

High priority feature consisting of multiple choice, true/false and short answer questions regarding Harry Potter trivia. These questions will be stored in a SQLite database.

3.4.2 Stimulus/Response Sequences

Users will have to answer a question to gain access to a room.

3.4.3 Functional Requirements

REQ-1: Must include multiple choice, short answer, and T/F at minimum.
REQ-2: Must scrub user input.

3.5 Database

3.5.1 Description and Priority

Holds all trivia questions/answers sorted by Hogwarts house along with save game board states.

3.5.2 Stimulus/Response Sequences

Each time a question is prompted, the database will retrieve it and compare the users answer to the stored correct answer. Users will also be able to insert trivia questions into the database. Save and loaded game boards will also be inserted and retrieved from the database.

3.5.3 Functional Requirements

REQ-1:	Must contain question tables for each Hogwarts House and a table for saving/loading.
REQ-2:	Each question table must include multiple choice, short answer, and T/F at minimum.
REQ-3:	Must scrub user input when inserting questions into database.

4. External Interface Requirements

4.1 User Interfaces

The system will be accessed via GUI through Visual Studio WPF.

4.2 Software Interfaces

The system will use a SQLite database to store trivia questions and game save information.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Will only be able to run on Windows machines due to Visual Studio WPF being limited to Windows Operating Systems.

5.2 Security Requirements

All user input received by the program must be scrubbed to aid in security.

6. Program Limitations

6.1 Feature Limitations

This section describes any features that we were not able to get fully functional.

6.1.1 Inserting Questions into Database Limitation

Currently our maze assigns a specific question to each room and there are only 15 rooms, so inserting questions into the database with ID's past 15 will never be used.

6.1.2 Unit Test Limitation

We were not able to create as many unit tests as we would like. Some of the methods in the MainWindow class were hard to test due to them being triggered by GUI events.

6.1.3 Installer Limitation

We were not able to get the installer to run our program at full functionality. The save/load game function and sound effects do not work. We tried using many different installers to fix it, but ultimately, we could not get it working.