Software Requirements Specification

for

Maze

Version 1.3 approved

Prepared by Logan Earl, Svetlana Kozubenko, and Pavel Trubchik

DAbuggers

11/3/2019

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 1

2.1 Product Perspective 1

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 GUI 3

3.2 Maze 3

3.3 Save/Load Games 5

4. External Interface Requirements 6

4.1 User Interfaces 6

4.2 Hardware Interfaces 8

4.3 Software Interfaces 8

4.4 Communications Interfaces 8

4.5 Performance Requirements 8

4.6 Safety Requirements 8

4.7 Software Quality Attributes 8

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Logan Earl | 12/10/19 | Finalization | 1.3 |
| Pavel Trubchick | 12/8/19 | Feature Updates | 1.2 |
| Svetlana Kozubenko | 12/5/19 | Updated Database Schema | 1.1 |
| Logan Earl | 11/3/19 | Initial Version | 1.0 |

# Introduction

## Purpose

The maze project’s goal is to create a computer game where a virtual world is traversed by answering questions and solving riddles. The game will be savable, meaning that you do not have to complete the game in a single sitting. Progress is stored by means of serialization and a SQLite database. All questions and answers will be stored in the SQLite database.

## Document Conventions

This document is designed to be flexible, and not overly regulated. However, if there are pieces of information that should be emphasized, it is recommended that they should be underlined. Titles and heading should be written in 18pt bolded font, subtitles in 14pt bolded font.

## Intended Audience and Reading Suggestions

Due to the unique nature of this piece of software, it is not necessarily designed for the layman to easily read through. Some experience with software and software design is recommended. This is acceptable in this case as the client that this software is designed for is very versed in the subject. This allows developers to be more upfront and technical regarding software specifics than would otherwise be possible.

## Project Scope

This project is intended to have a limited scope. It will be a one-off piece of software will little to no need to maintain it after release.

## References

There are currently no references for this project

# Overall Description

## Product Perspective

This project is designed to fulfill the requirements posted by Thomas Capaul, professor for CSCD350, from which this project was assigned. It is designed to be a stand-alone system with some basic support for manually sharing information between other instances of this software.

## Product Features

This software features

* A GUI and integrated hotkeys for ease of use
* A maze to navigate through the use of answering questions and riddles
* A way to save and load your games progress
* A way to create your own custom questions for use in future world generation
* A mass export/import feature for sharing large collections of questions to others in a human-readable format

## User Classes and Characteristics

Anyone with access to a computer should be able to install and play the game. Although the game is designed to be playable by anyone, it is mostly tailored toward those who enjoy trivia and puzzle games. The application will not differentiate users in any way, thus all users will have access to every part of the game.

## Operating Environment

The software currently only supports running on a Windows 7+ machine and requires that JRE (Java Runtime Environment) is installed. Mac and Linux machines are currently not supported, although due to the nature of JAVA only minor codes changes should be required to get the software running on other operating systems with JRE installed.

## Design and Implementation Constraints

All questions generated by the user must follow one of three templates in order to be savable in the SQLite database we have created for the purpose. These formats are as follows

True/False: Questions in this format have a text prompt and can either accept a true or false value as an answer. To answer, players are presented these two options and their input is compared against a desired result to determine correctness.

Multiple Choice: Questions in this format have a text prompt with 4 possible responses. To answer, players are presented with all four options. Upon choosing one, their input is compared against the desired result to determine correctness.

Short Answer: Questions in this format have a text prompt and a collection of keywords. Players are presented with the prompt and a text entry field. To answer, they must enter an answer into the field under a certain character limit. Their answer is searched for the presence of certain keywords that approximate the correct answer to determine correctness.

## User Documentation

This application will contain an intuitive GUI that will eliminate most need for help. In the case that some is needed however, the application will include a list of controls accessible from the main menu.

## Assumptions and Dependencies

This software has minimal dependencies. All that is required is the operating environment as specified in section 2.4.

# System Features

## GUI

3.1.1 Description and Priority

A GUI utilizing Java Swing components will be used to allow the player not easy and quick navigate through all menus and the world. The gameplay graphics will utilize 2D graphics, which will show the player and world from a top down view.

This is a High priority feature, as the GUI provides the user a way to interact with the game.

3.1.2 Stimulus/Response Sequences

The application will load to a main menu where the player will be able to start a new game or load an existing game. The main menu will also provide other buttons for other required functionality, including managing questions and enable cheats. Once the user starts or loads a game, the graphics panel will be loaded allowing the player to start playing the game.

3.1.3 Functional Requirements

REQ-1: Support for mouse and keyboard input in all menus.

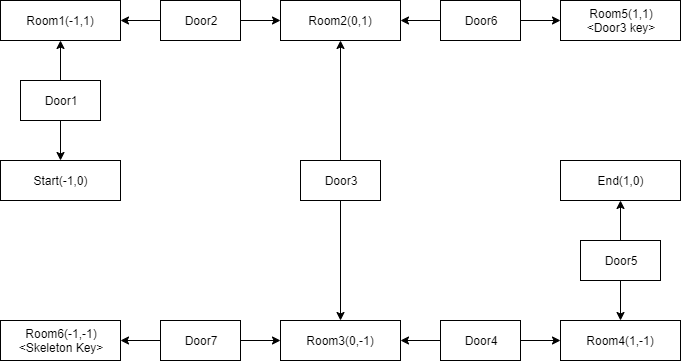
REQ-2: Keyboard input will be utilized for world navigation.

REQ-3: Must be responsive with minimal lag.

## Maze

3.1.1 Description and Priority

The maze is a collection of rooms. Each room may have one or more doors that connect that room with other rooms. Each door has a question associated with it. In order to pass through the door either it’s associated question must be answered, or it must be opened with a key item. Key items can be found while exploring the maze. There will be one or more skeleton keys placed in the maze as well. Skeleton keys are just like any other key, except that they can open any door. All items are destroyed when used except the master key, which is an administrator tool. An example maze might be structured like so



Questions can be attempted only once. A failed attempt to answer a question will result in the locking of said door. At this point, the user can still progress if they find the associated item for that door, or use any skeleton keys they may possess.

Wrapped up with this feature is the ability to procedurally generate mazes. This promotes the replayability of the project

This feature is core to the functionality of the project. Thus, on a scale of 1 to 10, this feature is a 10.

3.1.2 Stimulus/Response Sequences

Players begin in the starting room and must progress toward the end room. If they render a door impassible by failing to answer its question, and they have no item to open the door, and they lack the means to get to any more items, the game ends in which a game over screen is presented.

Otherwise, the player can choose a direction to move in (north south east west), the game will check for a door in that direction. If there is a door, and the door is opened the player simply moves to the desired adjacent room. If the door is not opened and there are question attempts remaining, the player is prompted with the question. The player can either choose an answer or utilize an item to bypass the question. In which case, the door is permanently opened as if they had chosen the correct answer. As the player enters a room, the room is checked for any items. If there are items, they are added to the player’s inventory. The player can hold any number of items, so there is never any reason to leave them behind.

Eventually, the player happens across the end of the maze, or renders the maze unsolvable. At this point the game ends by displaying either a winning screen or a game over screen.

3.1.3 Functional Requirements

When a world is created, the following requirements must be met

REQ-1: The world must be solvable

REQ-2: The world must not contain duplicate questions

REQ-3: The game must only end if there is no possible way for the player to win given the current state of the maze

REQ-4: The world must contain a minimum of 10 rooms including the start/end rooms

REQ-5: The world must not be trivially solvable. This means that if a solution exists, it must involve making choices between alternate routes. For an example, placing the exit a single door from the start is considered trivially solvable

## Save/Load Games

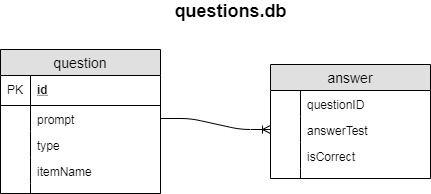
3.1.1 Description and Priority

When progressing through the game it is important to have the ability to save and load the current state of the maze to a file. This is primarily accomplished through the use of object serialization. This applies to all game objects excluding the game questions. These are already stored in the database, making it unnecessary to store them again via serialization. Instead, the serialized Door objects will contain integer question ids. When loading a game, the database will be queried to retrieve game questions.

The questions in the database will be stored in the following format:

Currently there are two tables in our database named questions.db. It has two

tables. One for storing questions and the second for storing answers. The description of our current schema is as follows:



Tuple will be added/removed from database tables with add/remove methods using SQL statements. The import feature with be used as a tool/aid in performing tuple insertion.

This feature is prioritized as an 8. It is blocked by the existence of a data model, and therefore must be completed once the maze feature is already complete.

3.1.2 Stimulus/Response Sequences

TBD. Waiting on some UI design

3.1.3 Functional Requirements

REQ-1: The world must be quickly savable, taking less that 5 seconds to save or load

REQ-2: Loaded worlds must be identical to their saved counterparts

# External Interface Requirements

## User Interfaces

The GUI utilizes Java Swing components for all menus and gameplay. The GUI will utilize a class extending JFrame, which will consists of various classes extending the JPanel class for various game screens. Here are some of the major panels/screens worth noting:

1. The main menu will include:

* New game button
* Load game button (proceeds to save selector panel)
* Master key toggle button
* View controls button
* Manage questions button (proceeds to question editor menu panel)

1. The graphics panel will only display the graphics for the game without any other controls.

Graphics panel will have the following key and mouse bindings:

|  |  |
| --- | --- |
| Esc | Open pause menu |
| W | Move up |
| A | Move left |
| S | Move down |
| D | Move right |
| I | View inventory |
| Mouse Wheel | Zoom In/Out |

1. The pause menu will include:

* Resume game button
* Save game button (proceeds to save selector panel)
* Load game button (proceeds to save selector panel)
* View controls button
* Exit to main menu button

1. Save selector panel will include:

* 4 save slots buttons
* Cancel button

1. The gameplay question panel will include:

* Question text box
* Answer input/selection (multiple types for different questions)
  + Multiple choice, consisting of 4 buttons
  + True false, consisting of 2 buttons
  + Short answer, text box for keyboard input
* Submit button
* Use item button. (Takes user to item selector panel)
* Cancel button

1. Item selector panel will include:

* List box (filtered by if an item is applicable to the door/question)
* Use item button
* Cancel button

1. Question editor menu panel will include:

* Add new question button (Proceeds to question selector)
* Edit question button (Proceeds to question selector)
* Delete question button (Proceeds to question selector)
* Cancel button

1. Question selector panel will include:

* Search text box
* List box of questions (filtered by search text box)
* Continue button (proceeds to question editor, except when selection if for deletion)
* Cancel button

1. Question editor panel will include:

* Question text box
* Question type drop down box
* Associated item text box
* Answers list box
  + Answer column, containing the answer text
  + Correct column, for marking the correct answer. With the exception of the short answers questions, all answer added to a short answer question are a correct possible answer.
* Add answer button
* Edit answer button
* Delete answer button
* Set correct answer button
* Save question button
* Cancel button

1. The inventory panel will include:

* List box of all items in possession
* Close button

## Hardware Interfaces

This software requires the use of a keyboard and mouse. The keyboard is required for the user to navigate the world, which will only be allowed through pre-set key bindings. The keyboard is also required when a user needs to enter text into a text box, which includes answering short answer questions as well as adding and modifying questions through the question editor. A mouse is required for clicking buttons and/or selecting an item from a list. All panels (except the graphics panel) are navigated using buttons presses, which is only allowed through mouse clicks. The mouse wheel is also used to zoom the graphics in and out.

## Software Interfaces

This software is developed to be run on the Java virtual machine thus, although not officially supported, can theoretically run on any java capable system. However, it is officially supported on Windows 7+ only. The software utilizes a connection to a SQLite database and requires certain OS-specific drivers to make this connection. Saving game progress utilizes the process of serializing game objects and will require read/write access to the filesystem.

## Communications Interfaces

This software does not implement or require any form of internet connectivity. Other Nonfunctional Requirements

## Performance Requirements

The game must be quick to respond to all user input. World graphics should be updated quickly without any noticeable lag. Although graphical animations are not a requirement in the final version of this game, any optional animations which are added should not have a noticeable impact on performance and responsiveness of the game. Thus, the game should be able to run on machines with low end or old hardware.

## Safety Requirements

This product does not have any particular safety requirements. It does not utilize access to any sensitive data, any information stored was created by and is under the complete control of by the users. Thus, it is their responsibility to maintain the integrity of their own system.

## Software Quality Attributes

This software will have the following qualitied

* Flexibility: This software will be flexible to changing conditions
* Testability: This software will be designed with unit testing in mind and will lend itself to the process