Software Requirements Specification

for

Knowledge Labyrinth, Release 4.0

Version 4.0 approved

Prepared by Johnathan McNutt

Team Rapidash Cachers

10 July 2015

Table of Contents

Revision History

Name	**Date**	**Reason For Changes**	**Version**
Johnathan McNutt	4/14/15	Initial draft	1.0
Johnathan McNutt	6/10/15	Updated SRS for final delivery	4.0

1. Introduction

1.1 Purpose

This SRS describes all requirements for release 4.0 of our game "Knowledge Labyrinth". This document is intended to be used by the members of the project team that will implement and verify the correct functioning of the system.

1.2 Document Conventions

Unless otherwise noted, all requirements specified here are high priority and committed for release 4.0.

1.3 Intended Audience and Reading Suggestions

The intended audience is the development team for the project. The formatting of the SRS follows the structure of the Table of Contents above. The reading order for this documentation is suggested to be sequential or reading sections on an as needed basis.

1.4 Project Scope

With this project we intend to write a program that generates a maze that the player can traverse and answer trivia questions in order to eventually finish the game or fail enough questions to make victory impossible. Further information can be found in the Trivia Maze requirements provided by the instructor.

1.5 References

See our team coding standards for more information on code requirements.

2. Overall Description

2.1 Product Perspective

This project is self-contained.

2.2 Product Features

The project will contain three main features: graphics interface, maze generation, and database management.

2.3 User Classes and Characteristics

Our program is designed as a game and intended for the purposes of recreation. No prerequisite is required in order to play the game.

2.4 Operating Environment

The program will be using C# 5.0 with SQLite x32 architecture. The program will run on all operating systems that support both.

2.5 Design and Implementation Constraints

Locally created and maintained SQLite databases will be stored with each copy of the program. The programs link to the databases as well as the user's ability to manipulate the database is a potential security risk since it gives the user the ability to modify their own experience.

2.6 User Documentation

Button located on startup will allow the user to check the goal and the controls of the game. No further user documentation will be included.

2.7 Assumptions and Dependencies

At this point no assumptions are being made.

3. System Features

3.1 Graphics & Movement System

3.1.1 Description and Priority

This feature creates and displays basic graphics and allows the user to move the player character along floors but does not let them go out of bounds.

3.1.2 Stimulus/Response Sequences

Activating the feature is done by click a "graphics/movement demo" button which opens a new window displaying the gameplay demo. Using arrow keys while the new window is active will move the blue player along the flooring. When the user is done with the demo they may close the new window using standard window closing techniques.

3.1.3 Functional Requirements

No special requirements are needed for this feature.

3.2 Maze Generation

3.1.1 Description and Priority

This feature includes the ability to generate a map that includes a grid of room and corresponding passageways leading between them. Each path will be randomly generated upon start up, unless loading in a previous map.

3.1.2 Stimulus/Response Sequences

Maze Generator will be called on when a new game is started. It will create a 16 room maze by default

3.1.3 Functional Requirements

No requirements are needed. All functionality is handled inside the methods, with no opportunity for bad input.

3.3 Database Management System

3.1.1 Description and Priority

The Database Management System is a system build to manage and manipulate the databases for use within the game. The system will help by providing a means to easily add questions and pull them into the game itself. It could also be prepared to allow players to add or remove new questions to the game.

3.1.2 Stimulus/Response Sequences

The system can be activated at two points during the program's operation. The main situation which the system is used is to load a random question from the database to use during gameplay. The other situation is to manipulate the questions within the game itself. The System will handle the queries automatically, so the user of the program will never have to make a query statement themselves.

3.1.3 Functional Requirements

REQ-1: The main requirement is a DLL to support interactions with SQLite databases.

REQ-2: Protection from SQL Injections is also required in order to prevent changing the state of database so the program can reliably import questions from the database.

4. External Interface Requirements

4.1 User Interfaces

The title screen will have four buttons for the user to pick from. These will include the New Game, Load Game, Directions, and Database Manipulation buttons. When New Game or Load Game buttons are clicked it will bring up another window with the main interface that allows the user to move the character around the map and navigate the maze. The Database Manipulation button will bring up another interface that allows the user to enter in new questions, create new databases and remove questions.

4.2 Hardware Interfaces

All user interactions will be handled throw keyboard and mouse inputs. No extra peripherals are required.

4.3 Software Interfaces

Each team member will use Visual Studios 2013 on a Windows 7 or 8 operating system. All code will be done in C# 5.0 or later using a SQLite x32 database. All libraries used will be standard to Visual studios and C# with the exception of SQLite specific libraries.

4.4 Communications Interfaces

All program content is stored locally and requires no networking.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The low cost of our programs operations makes it unlikely that any system would have trouble running our game at a smooth 60 frames per second.

5.2 Safety Requirements

All program interactions are self-contained and do not effect outside systems.

5.3 Security Requirements

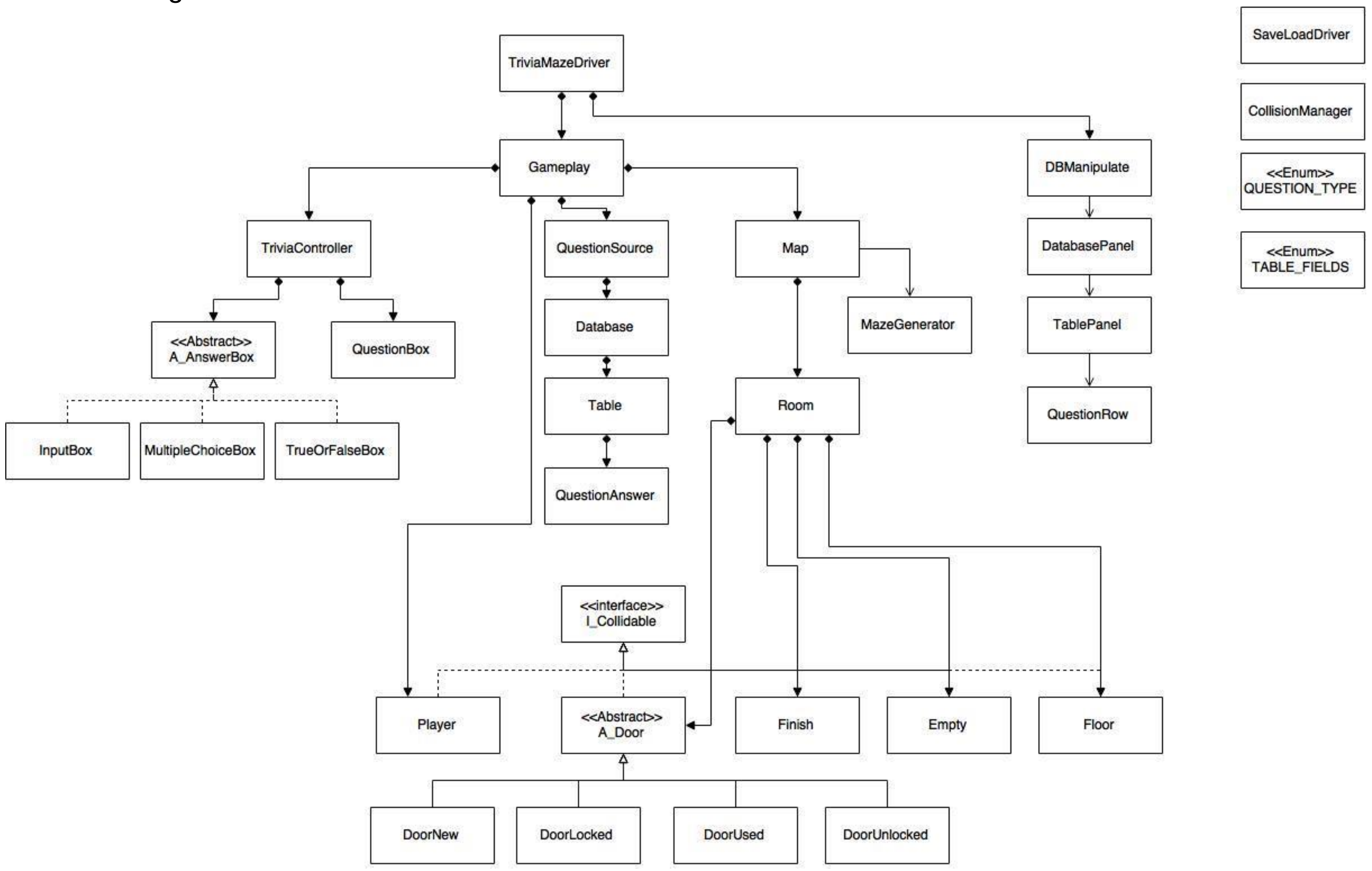
Users who wish to manipulate the local database will be allowed to. Anyone who wishes to manipulate the database outside of the program will be able to however, since it is local it is at their own discretion. There are no personal security risks as no user data is stored by the program.

5.4 Software Quality Attributes

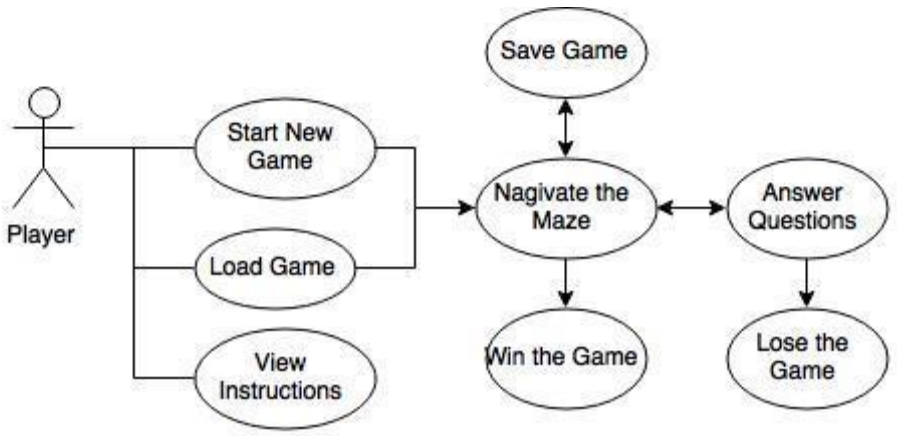
When designing the GUI for this program, we are designing it with ease of use in mind. This lets the user intuitively use the program instead of having to learn to use the program. Additionally, the user is enabled to add in more databases at their leisure.

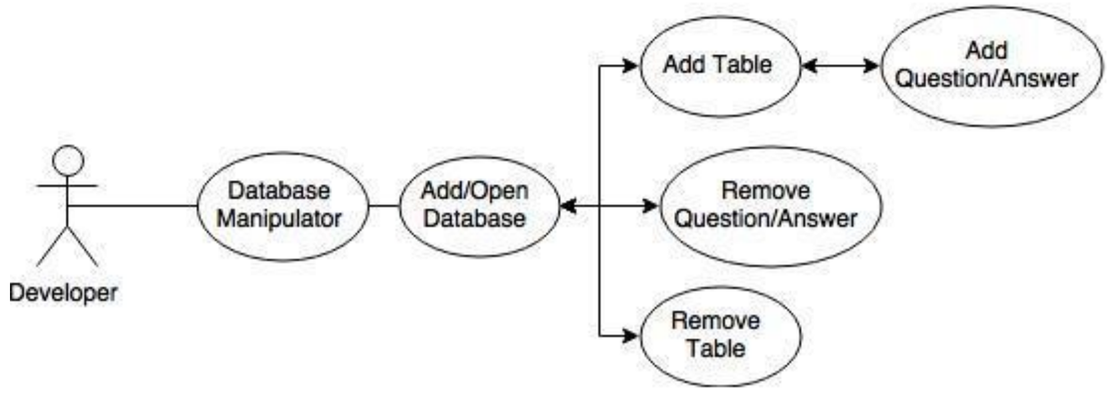
Appendix A: Analysis Models

Class Diagram:



Use Case Diagram:



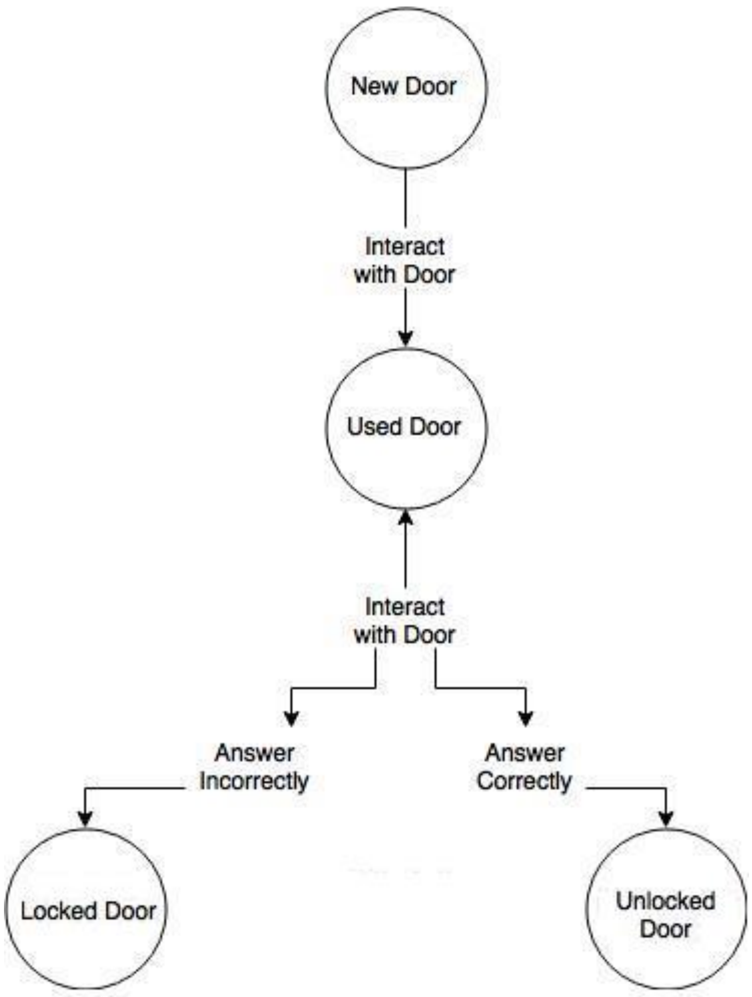


Use Case Description:

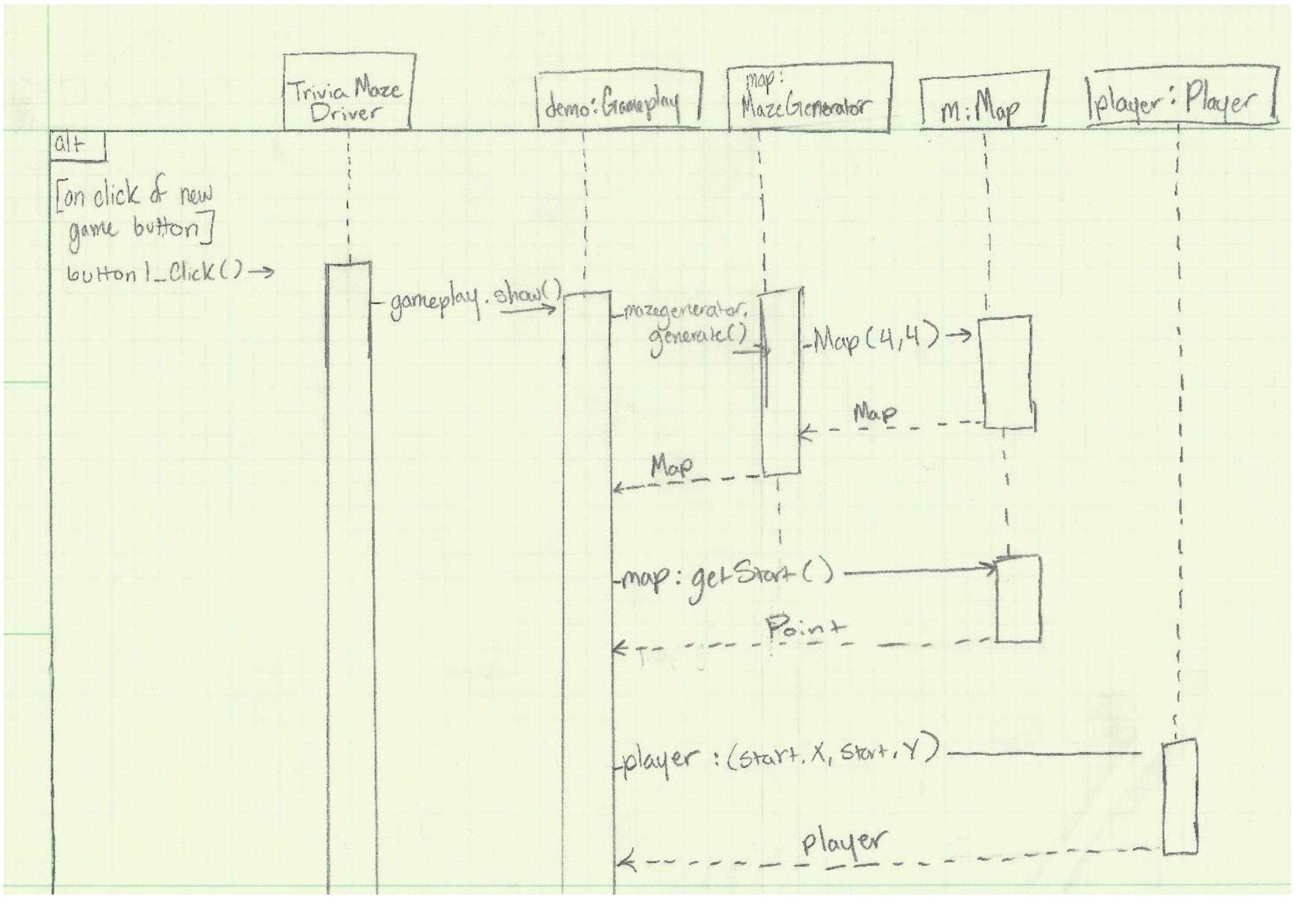
Use Case Name	Playing the Game
Participating Actors	Player
Flow of Events	1. Clicks on the New Game button 2. Navigates Maze 3. Answers Questions 4. Makes it to finish
Alternative Flows	1. a. Loads Saved Game 1. b. Clicks on Directions 2. a. Saves Game 2. b. Exits Game 4. a. Loses Game
Entry Conditions	User successfully installs and starts up the game
Exit Conditions	User beats, loses, or exits the game

Use Case Name	Adding Questions to the database
Participating Actors	Developer
Flow of Events	1. Clicks on Database Manipulation Option 2. Opens a Database 3. Add Table 4. Add Question and Answers
Alternative Flows	2. a. Add Database 3. a. Remove Table 4. a. Remove Question and Answers
Entry Conditions	Developer successfully installs and starts up the game selecting Database Manipulation on the main menu
Exit Conditions	Exits the program

State Diagram:



Sequence Diagram:


Trivia Maze Driver
demo: Gameplay
map: MazeGenerator
m: Map
player: Player
alt
[on click of new game button]
button1_Click()
gameplay.show()
mazegenerator.generate()
Map(4,4)
Map
Map
map: getStart()
Point
player: (start.X, start.Y)
player