System Requirements

Specification

The Amazing Maze

Brought to you by:

**IDK**

Sarah Headley

Giang Bui

Tony Moua

**Table of Contents**

1. **Introduction**
   1. Purpose
   2. Scope
   3. Document Conventions
   4. Major Constraints
2. **Overall Description**
   1. Product Perspective
   2. Product Functions
   3. User Characteristics
   4. Assumptions and Dependencies
3. **System Features**
   1. Exit the maze
      1. Description and Priority
      2. Stimulus/Response
4. **External Interface Requirements**
   1. User Interface
   2. Hardware Interface
   3. Software Interface
   4. Communications Interface
5. **Other Nonfunctional Requirements**
   1. Safety Requirement
6. **Validation Criteria**
   1. Tests
7. **Models and Description**
   1. Data Objects
   2. Sequence Diagram
   3. State Diagram
   4. UML
8. **Introduction**
   1. **Purpose**

The purpose of this document is to provide a detailed description and requirements for the Amazing maze software. It will also explain the interactions between user and software.

* 1. **Scope**

The “Trivial Maze” is a software that will allow users to travel through a 2-diminsional maze that contains an entrance (where users starts the game), an exit (where user will win the game when exit point is reached) and rooms containing doors between the entrance and exit where you will have to make your way through by answering questions correctly.

By answering questions correctly the door you have chosen to go through will open. If answered incorrectly the door will lock and user will not be able to open the same door again. You have the option to skip the question (up to 3 questions) but the door will not open for you. If all doors that surround the rooms are locked the game will be over. User may back track their way back to previous rooms through the door that was unlocked. Doors will forever stay locked or unlocked depending if the answer is correct or incorrect.

* 1. **Document Conventions**

|  |  |
| --- | --- |
| DB | Database |
| T | True |
| F | False |
| MC | Multiple Choice |

**1.4 Major Constraints**

No Major constraints as of now

1. **Overall Description**

**2.1 Product Perspective**

This software will engage the user with choices to transverse through a maze upon answering questions to unlock doors to the next room.

**2.2 Product Functions**

With this software it will begin by prompting the user for a name. Once name has been entered software you will allow you to choose a level of difficulty, easy, medium and hard. You will then be placed in a room starting at coordinates (0,0); location coordinates will be visible for users Once door is selected, users will be prompted with a randomized question. If answered correctly then the door will open, if answer incorrectly the door will lock and you may never open it again for the remaining duration of the game. You win when you reached the exit and lose when all doors leading to the exit are locked.

The game will print out a map every time you answer a question right or wrong. Answer the question with something that is not specified and the game will keep asking you the question.

: (colon) – the door to the room is opened.

# (pound) - the door to the rooms is locked.

**2.3 User Characteristics**

There are two types of users that can interact with this software: user of the game and administrators.

The user playing the game will only be able to play the game as directed. This means the user has to be able to choose a difficulty level, enter a user name, transverse through the maze, choose doors, answer the question that is randomly given by the questionnaire DB, retrace user path if user decide to go back through the unlocked doors and allow user to win or lose depending on the state of where they are at.

The administrators will interact with the software directly if software malfunctioning occurs, updates, add and remove questions and do what is necessary for the game to run smooth.

**2.4 Assumptions and Dependencies**

Software is for all ages.

Software question DB will need to be updated every so often for an enjoyable game play.

1. **System Features**

**3.1 Exit the Maze**

**3.1.1** Description and Priority

User must answer the question correct to move around the maze to find the exit. If question is answered incorrectly then door will lock and cannot be open again for the reminder of the game. Priority = High.

**3.1.2** Stimulus/Response

Stimulus: picks a door inside the room

Response: System then connects to database to retrieve a random question (T/F or multiple choice)

Stimulus: Answers question correct.

Response: triggers the door connected to the question to open

Stimulus: Move to next room.

Stimulus: Answers question incorrect.

Response: triggers door to lock for the rest of the game

1. **External Interface Requirement**

**4.1 User Interface**

Once you have installed the software to your system, open up the application and you should be prompted to enter a name. There will be no save. Your progress will reset everything you win, lose or decide to quit the game.

After entering username you will then be prompt to choose a difficulty level ranging from easy, medium and hard. These levels will indicate how big the size of the maze will be. Now you will begin your quest to find an exit within the maze. To do so, every door will contain a random question, true/false or multiple choices, if answered correctly the door will open, if answered incorrectly the current door will lock.

**4.2 Hardware Interface**

No hardware interfaces have been identified

**4.3 Software Interface**

**4.4 Communication Interface**

**4.4.1** Trivial software will connect to the questions DB to get a new question once a door has been triggered.

**4.4.2** Trivial software will not save users progress.

1. **Other Nonfunctional Requirements**

**5.1 Safety Requirements**

No safety requirements

1. **Validation Criteria**

**6.1 Test Cases**

The test cases that we have included in our test is to check for doors, out of bounds, are there enough questions, does the answer comply, a valid path and to check that the column and row are correct.

To test for doors we have to make sure that the player will not go out of bounds. Make sure that it’s a door and not the out of bounds line

We need to make sure that there are questions in the database at all times. If one table of question runs out then the software will ask for another question from the table that has questions.

Make sure that the DB will send the correct answer to the mazes to compare with the answer that the player gives us. If we test NULL then the DB is not sending the answer to the maze class to be compared.

Make sure that there is always a valid path. If valid path complies then continue the game. If there’s not a valid path then make sure the software knows that the game is over.

Make sure that when the player choses his/her size for the maze, that the rows and columns get their size. If one is off then the test will let us know that something is wrong with the columns and rows.

1. **Models and Descriptions**

**7.1 Data Objects**

Game class: Main will be held in the game class. This class will also prompt the user for a name and communicate with the maze class.

Maze class: Maze class will communicate with the game class, door class, room class and the database fill with questions. Between maze and door class, the maze will want to know what door the user wants to go into. Once that is known the maze will communicate with the room class to get the room. The question will then be granted to the user for the user to answer. Once an answer is entered, maze class will go back to the room and grab the answer for the room. Room will talk to the DB and the answer will be compared the users answer.

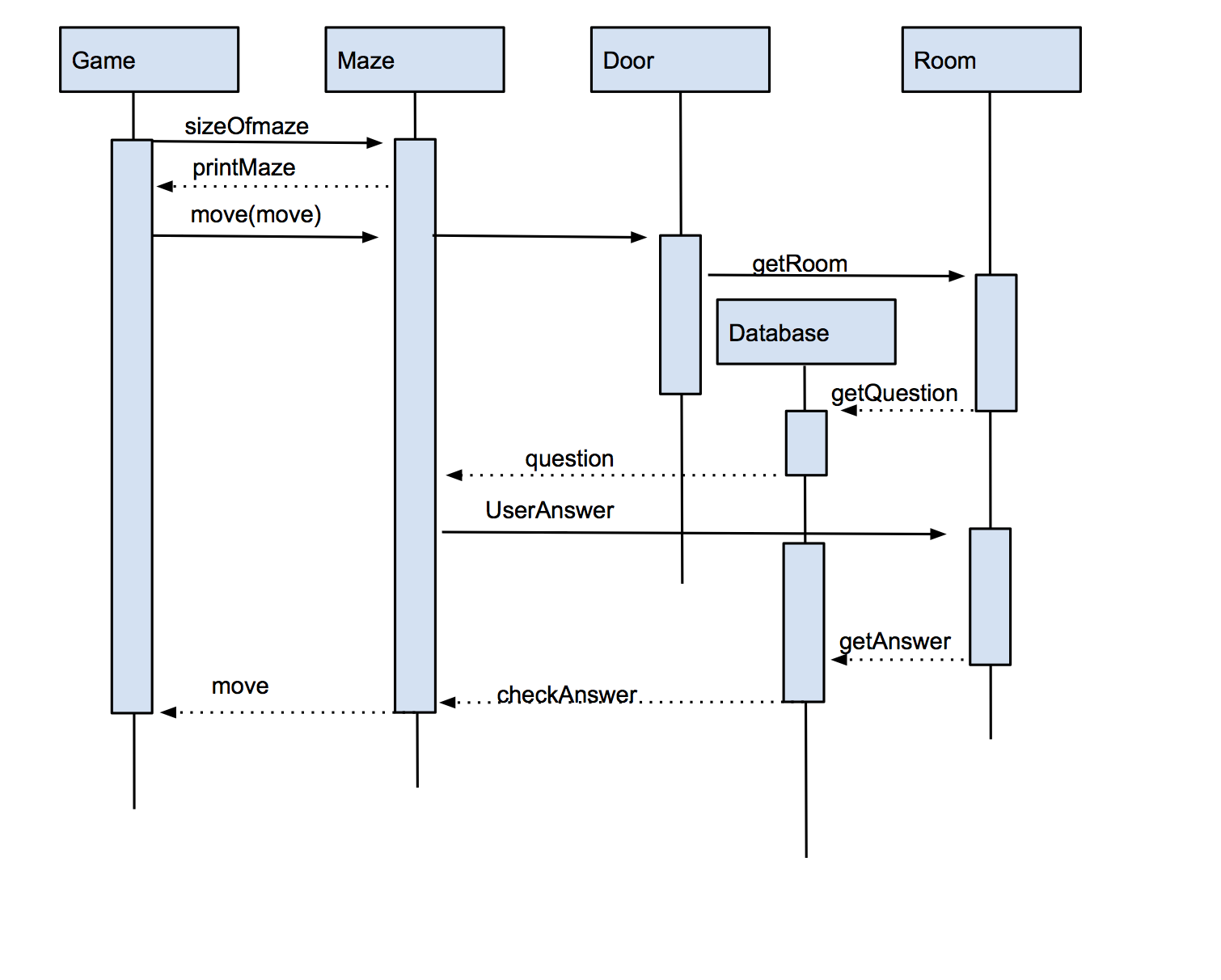
Door class: This class will trigger a question after the user has chosen the direction. It will get a question and get the answer for the question and compare it with the users answer.

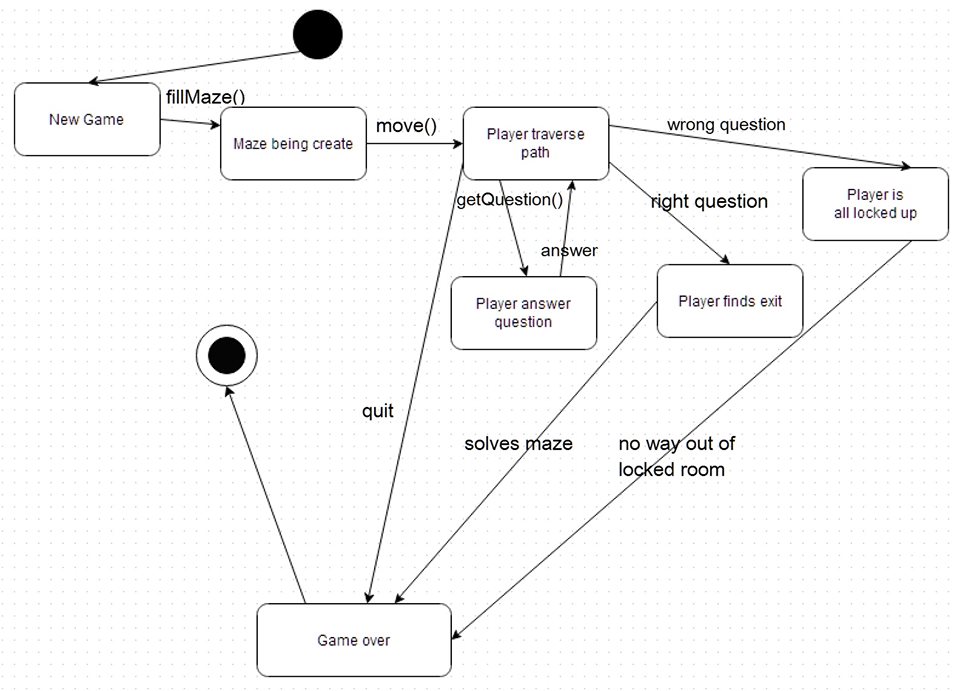
Room class: this class will get the room you want to go to.

**7.2 Sequence Diagram**

The game will ask the user how big the user wants his/her maze. After the size is chosen the maze class will build the maze and send it back to the game class. Once the game gets a hold of it then the game will print out the maze. Next, the game will ask user which direction the user wants to go, right, left, up or down. Once a direction is chosen, the game class will talk to the maze class to move. Maze will then talk to the door class and the door class will talk to the room class to get the room that you want to be in. Once the room is acquire, it will ask the DB for a question. The question is then sent back to the maze to display the question to the console. An answer is wanted by the console from the user and once the user has enter a answer, the answer will get sent back to the room class so the console can get the answer for the question in that room you want to travel to. Once the answer is retrieved from the DB it goes back to the maze class to check and compare both answers. If they are equal then that door to that room opens. If it is not equal then the door is looked fro the entire game.

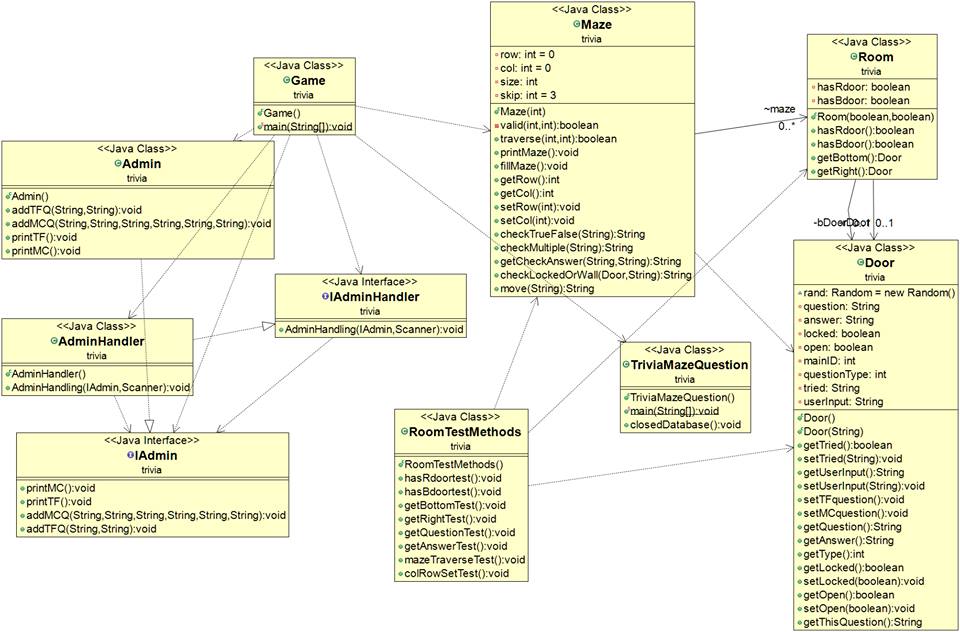
Repeats itself till you get to the end of the maze and win or you lock all your paths to the exit and lose.



**7.3 State Diagram**

**State diagram shows you what state you are in and what it delivers to the next state. As you see in the diagram above, user will start a new game every time they launch the application. New game will prompt the user for a name, and ask for a size the user wants. Next the maze will be created. The player will transverse and make its way through the maze by answering questions. The player must answer the question once question is answered, if answered correctly player finds exit. If player answers wrong, door will be locked. The game is over if the player quits, solves the maze or locks up all the doors to the exit.**

**7.4 UML**

****