Software Requirements Specification

Amazing Trivia Maze

Austn Attaway aaa2000@uw.edu
Chau Vu cvu1@uw.edu
Daniel Jiang djiang7@uw.edu

Created: May 4th, 2021
Last Updated: June 9th, 2021

Table of Contents

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document describes the functional and non-functional requirements for our Amazing Trivia Maze game made for TCSS 360. This is intended for the project development team and clients as a reference to ensure that the project meets all functional and non-functional requirements.

1.2 Definitions, Acronyms, and Abbreviations (Document Conventions)

Term/Abbreviation:	**Definition:**
ATM	Amazing Trivial Maze
Player	The person that is playing the game

1.3 Intended Audience and Reading Suggestions

This SRS document's intended audience are developers and product managers for the Amazing Trivia Maze game.

- Section 2 provides an overall description of the Amazing Trivia Maze program
- Section 3 covers the system features and is used for planning iterations
- Section 4 contains external interface requirements
- Section 5 covers all the other non-functional requirements not previously mentioned

1.4 Project Scope

The ATM game is a trivia maze game in which the Player starts at a particular location in the maze and travels through the maze by answering trivia questions, moving room by room. The theme of the user interface and the questions is popular video games. We hope the software will be entertaining, interesting, and mildly difficult for someone who has a general familiarity with video games.

1.5 References

Sprite Sheet/Imagery:
Sprite sheet
Image resizing

Movement:
Player sprite movement

Audio:
Menu background music
Gameplay background music
Winning game sound effect
Losing game sound effect
Door opening sound effect
Losing brain sound effect

Font:
Title font
Text font

2. Overall Description

2.1 Product Perspective

The ATM game is a standalone game that allows the Player to navigate the maze and answer trivia questions through keyboard and mouse interactions. The game will have a graphical user interface (GUI), a model and controller to manage the data the GUI represents, and a SQLite database to store trivia questions the Player will answer.

2.2 Product Features

Starting the ATM game, the Player is first shown a “Start” game panel in which they can click the “Start” button to start the game. The Player will be inserted into a predetermined start room based on the maze map. From there on, the Player will be able to move their character using the WASD keys to move within and between each room. By hitting the spacebar next to a door, the Player interacts with the door and answers a question to unlock and move into the next room. If the Player eventually reaches the end room, they will be shown a “You Win!” screen and the Player can restart or quit the game. The Player will have up to 5 chances to answer the questions correctly and reach the end room before they lose the game. If they lose the game, they are shown a “You Lose!” screen and the Player is allowed to restart or quit the game.

2.3 User Classes and Characteristics

There is a single type of user for the ATM game, which is the person playing the game (the Player).

2.4 Operating Environment

This system is compatible with the Mac BigSur Operating System and the Windows 10 Operating System. The trivia questions database our system uses SQLite and the .jar file for that is within the system files. The Java Runtime Environment (JRE) that this application can run on is Java 14.0 and above.

2.5 Design and Implementation Constraints

The design of our system uses a SQLite database to store all of the maze's trivia questions that the Player will answer. The language our trivia questions are in is English. Our system is developed in Java and uses Java's Swing library and the code is written to conform to the University of Washington Tacoma's coding conventions.

2.6 User Documentation

Not applicable at the time.

2.7 Assumptions and Dependencies

There are no other assumptions and dependencies that have not already been stated in Section 2.4 Operating Environment.

3. System Features

3.1 Before the Game

3.1.1 Start Game Title Screen and Start Button

ID: SF 1.1

3.1.1.1 Description and Priority

When the Player starts the program, the first thing they will see is a start screen that contains a start button and a title background image. When the Player presses the start button, a new game instance will start.

Priority: Medium

3.1.1.2 Stimulus/Response Sequences

S1: The Player invokes the program and starts to run it.

R1: A GUI frame pops up that contains the title screen and the "Start" button and the main menu background music starts playing.

S2: The Player clicks the start button with their mouse.

R2: An new instance of the game begins and the main menu background music stops playing and the gameplay background music begins playing.

3.1.1.3 Functional Requirements

REQ-1: A frame is required for containing the start panel.

REQ-2: A way to interact with the panel with the mouse and determine if it is over the button when it is clicked.

REQ-3: A background image that will be displayed on the panel.

REQ-4: A way to indicate that the start button is being hovered over.

3.2 Navigating the Maze During Game

3.2.1 Minimap

ID: SF 2.1

3.2.1.1 Description and Priority

As the Player traverses the maze and unlocks rooms, a minimap in the upper left corner of the frame is visible that represents the already visited rooms thus far.

Priority: Low

3.2.1.2 Stimulus/Response Sequences

S1: The Player enters a room that has not been in before.

R1: The minimap will update to include this room and the path between it.

S2: The Player enters a room of any type.

R2: The minimap indicates with a different color where the Player is currently located.

3.2.1.3 Functional Requirements

REQ-1: Access to the matrix of rooms so we can use their state to decide whether or not to draw each room.

REQ-2: A way to indicate whether or not a given room has been visited.

REQ-3: A way to indicate where the Player currently is in the maze so we can draw that in a different way.

REQ-4: Ability to access whatever we want to draw on.

3.2.2 Character Sprite Movement, Animation, and Interaction

ID: SF 2.2

3.2.2.1 Description and Priority

The Player can use WASD keys to move the character sprite up, left, down, and right respectively throughout the rooms in the maze. The sprite animates in the correct direction to wherever the character is moving. The Player can use the Spacebar to interact with doors and items if they are nearby.

Priority: High

3.2.2.2 Stimulus/Response Sequences

S1: The Player presses on the W key.
R1: The character sprite faces and moves upwards.

S2: The Player presses on the A key.
R2: The character sprite faces and moves leftwards.

S3: The Player presses on the S key.
R3: The character sprite faces and moves downwards.

S4: The Player presses on the D key.
R4: The character sprite faces and moves rightwards.

S5: The Player presses on the W and A keys.
R5: The character sprite faces and moves up and left.

S6: The Player presses on the W and D keys.
R6: The character sprite faces and moves up and right.

S7: The Player presses on the S and A keys.
S7: The character sprite faces and moves down and left.

S8: The Player presses on the S and D keys.

R8: The character sprite faces and moves down and right.

S9: The Player releases the W key.

R9: The character sprite stops moving in the upwards direction.

S10: The Player releases the A key.

R10: The character sprite stops moving in the leftwards direction.

S11: The Player releases the S key.

R11: The character sprite stops moving in the downwards direction.

S12: The Player releases the D key.

R12: The character sprite stops moving in the rightwards direction.

S13: The Player presses on the Spacebar.

R13: The character sprite interacts with a door or an item if the character sprite is next to and facing the door or item. The sprite also stops moving in all directions.

3.2.2.3 Functional Requirements

REQ-1: A way to know when a particular key (WASD) or Spacebar is pressed.

REQ-2: A way to know when a particular key (WASD) or Spacebar is released.

REQ-3: A particular animation set for each direction of movement (in this case, the proper sprite in the sprite sheet).

REQ-4: A way to animate the Player by iterating through images in the animation sets.

3.2.3 Room Environment

ID: SF 2.3

3.2.3.1 Description and Priority

The frame contains the image that represents the room environment, available doors to navigate through, and items on the floor.

Priority: High

3.2.3.2 Stimulus/Response Sequences

S1: The door is locked.
R1: The door visual is in its locked state.

S2: The door is unlocked.
R2: The door visual is in its unlocked state.

S3: The door doesn’t exist.
R3: The door image is replaced with a wall.

S4: The Player answers a door’s question correctly.
R4: The door visual will change to its unlocked state.

S5: The Player uses an unlocked door to move into the next room.
R5: The door opening sound will play and the environment will update to represent the new room.

S6: The Player interacts with an item.
R6: The item disappears from the environment.

3.2.3.3 Functional Requirements

REQ-1: A way to know when the Spacebar is pressed.

REQ-2: A way to know whether a North, South, East, or West door exists in the current room.

REQ-3: A way to know whether or not the state of a door is locked or unlocked.

REQ-4: A way to know whether or not thePlayer sprite is close enough to a door to interact with it.

3.2.4 Heads-Up Display (HUD)

ID: SF 2.4

3.2.4.1 Description and Priority

On top of the environment background are icons that display valuable information to the Player. On the top right, a list of brains represent the number of incorrect answers the Player can have before losing the game. On the bottom left, a list of items represent the items that the user has/can have throughout the game.

Priority: High

3.2.4.2 Stimulus/Response Sequences

S1: The Player answers a question incorrectly and there are at least 2 brains.

R1: The lose brain sound will play and a brain gets removed from the right of the list of brains.

S2: The game answers a question incorrectly and the Player has 1 brain.

R2: The lose brain sound will play and the final brain gets removed and the game is over.

S3: The Player gets an item that increases their brains by 1.

R3: A brain is added to the right of the list of brains.

S4: The Player has not picked up a particular item on the item bar.

R4: That particular item is greyed out on the item bar.

S5: The Player has picked up a particular item that can be used to answer questions.

R5: The item’s color is shown on the item bar.

3.2.4.3 Functional Requirements

REQ-1: A way to know how many brains the Player has remaining.

REQ-2: An image to use to display the brains.

Item requirements are TBD

3.2.5 Door Interaction

ID: SF 2.5

3.2.5.1 Description and Priority

When the character sprite is near a door, the Player can use the spacebar to interact with the door. If the door is locked, the question frame is invoked. If the door is unlocked, the Player moves into the room that the door is connected to.

Priority: High

3.2.5.2 Stimulus/Response Sequences

S1: The Player interacts with a locked door.

R1: The question frame pops up.

S2: The Player interacts with an unlocked door.

R2: The door opening sound will play and the Player moves into the room the door is connected to.

3.2.5.3 Functional Requirements

REQ-1: A way to interact with a door using the space bar.

REQ-2: A way to know if a door is locked or unlocked.

REQ-3: A way to change rooms if the door is unlocked and interacted with.

REQ-4: A way to pop up a question frame if the door is locked and interacted with.

3.2.6 Item on Floor Interaction (Not yet implemented)

ID: SF 2.6

3.2.6.1 Description and Priority

The Player can interact with the item on the floor using the Spacebar. If the item is a question item, it is added to the item bar. If the item is a use-on-pickup item, it will be used when it's interacted with. If the item is the GOLDEN CONTROLLER and the user interacts with it, the game is won.

Priority: low

3.2.6.2 Stimulus/Response Sequences

S1: The Player interacts with a question item.

R1: The question item is added to the item bar.

S2: The Player interacts with a use-on-pickup item.

R2: The item's ability is used immediately.

S3: The Player interacts with the GOLDEN CONTROLLER which is in the final room.

R3: The game is won by the Player and the "You Win!" screen is invoked.

3.2.6.3 Functional Requirements

REQ-1: A way to know if an item is close enough to the Player to be interacted with.

Other item requirements TBD

3.2.7 Maze Options/Layouts

ID: SF 2.7

3.2.7.1 Description and Priority

A number of mazes will be built by the development team's maze builder. When a new game begins, the new game chooses one of the pre-built mazes at random that have not already been used in the current game session. If the Player has played all of the available maps in the current session, they will play one of the maps they already played.

Priority: High

3.2.7.2 Stimulus/Response Sequences

S1: The Player starts a new game (there are remaining mazes the Player has not played yet).

R1: A maze that has not been played yet gets picked for the new game.

S2: The Player starts a new game (there are no remaining mazes the Player has not played yet).

R2: A random maze is picked.

3.2.7.3 Functional Requirements

REQ-1: A way to start a new game.

REQ-2: A way to manage the available mazes that have already been used.

REQ-3: A way to set up a new state of the game with a new maze that has not been used.

REQ-4: A way to reuse old mazes in new game states if all mazes have been used.

3.3 Answering Questions During the Game

Question, answer, submit, cancel, brains, items section

3.3.1 Question

ID: SF 3.1

3.3.1.1 Description and Priority

The top section of the question frame contains the text based question prompt.

Priority: High

3.3.1.2 Stimulus/Response Sequences

None

3.3.1.3 Functional Requirements

REQ-1: A way to get the question prompt from the question currently being asked.

REQ-2: A panel to draw the question on.

REQ-3: A frame to contain the question panel.

3.3.2 Answer

ID: SF 3.2

3.3.2.1 Description and Priority

The section below the question represents the possible answers the Player can select. The possible types of answer sections could be multiple choice, true/false, or free response/short answer.

Priority: High

3.3.2.2 Stimulus/Response Sequences

S1: The current question is a multiple choice question.

R1: Each answer can be picked using a radio button that is on the left of the text.

S2: The current question is a true/false question.

R2: Each option (true or false) can be selected with a radio button.

S3: The current question is a free response/short answer question.

S4: A text box exists in which text can be written inside to represent the Player's answer.

3.3.2.3 Functional Requirements

REQ-1: A way to get all the possible options for a multiple choice question.

REQ-2: A way to get all the possible options for a true/false question.

REQ-3: A way to allow the user to select one of the radio button options.

REQ-4: A way to add a text box to a question panel that can be typed in.

3.3.3 Submit

ID: SF 3.3

3.3.3.1 Description and Priority

A submit button exists which allows the Player to submit the answer that they currently have selected or typed in.

Priority: High

3.3.3.2 Stimulus/Response Sequences

S1: The Player submits an incorrect answer.

R1: A brain is removed and the question frame closes and the lose brain sound plays.

S2: The Player submits the correct answer.

R2: The question frame closes, the door opening sound plays and door opens and the Player walks through.

S3: The Player has selected a radio button from either a multiple choice or true/false question and hits the submit button.

R3: The answer is graded and if it is correct, the frame closes and the door is opened. If it is incorrect, the frame closes, the lose brain sound plays and they lose a brain.

S4: The Player has not selected a radio button (for multiple choice and true/false questions) and hits submit.

R4: The question is answered incorrectly so the lose brain sound plays and the Player loses a brain.

S5: The Player hits submit for a free response question with the wrong answer.

R5: The lose brain sound plays and a brain is lost, the frame closes.

S6: The Player hits submit for a free response question with the correct answer.

R6: The frame closes, the door opening sound plays, and the Player moves through the door into the next room.

3.3.3.3 Functional Requirements

REQ-1: A way to check whether or not the correct option button is pressed.

REQ-2: A way to check whether or not the given string input matches the expected correct answer for a free response question.

REQ-3: A way to have the Player lose a brain when a question is answered incorrectly.

REQ-4: A way to move between rooms automatically when the frame is closed.

REQ-5: A way to close the frame once the question has been answered.

REQ-6: A way to shuffle the options after the frame closes.

3.3.4 Brains

ID: SF 3.4

3.3.4.1 Description and Priority

In the top right of the question frame a list of brain icons represent the number of incorrect answers the Player can have before losing the game.

Priority: Medium

3.3.4.2 Stimulus/Response Sequences

S1: The Player incorrectly answers a question.

R1: A brain is removed from the right of the brain list.

3.3.4.3 Functional Requirements

REQ-1: A way to know how many brains are remaining.

REQ-2: A way to display the brains.

3.3.5 Toolbar Items

ID: SF 3.5

3.3.5.1 Description and Priority

Below the answers and submit buttons, there is a section that provides information about and the ability to use question items. A dropdown that contains the items the user has found is at the top, with a "Use Item" button next to it. Below the dropdown and "Use Item" button, a name, image icon, and description of the currently selected item is displayed.

Priority: Medium

3.3.5.2 Stimulus/Response Sequences

S1: The Player has not selected an item from the dropdown.

R1: The "Use Item" button will be greyed out.

S2: A particular item is not unlocked by the Player yet.

R2: The option in the dropdown is greyed out and not pickable.

S3: The Player selects a question item they have from the dropdown menu.

R3: The name, image icon, and description of the selected item is displayed below the dropdown menu.

S4: The Player uses an item by hitting the "Use Item" button.

R4: The action the item does occurs and the item is removed from the dropdown list.

3.3.5.3 Functional Requirements

Item requirements TBD

3.4 After the Game

3.4.1 Win Screen

ID: SF 4.1

3.4.1.1 Description and Priority

When the Player has interacted with the GOLDEN CONTROLLER the frame changes to the “You Win!” screen that contains a “Play Again” button and an “Return to Main Menu” button, as well as some text for game credits.

Priority: Medium

3.4.1.2 Stimulus/Response Sequences

S1: The Player interacts with the GOLDEN CONTROLLER.

R1: The “You Win!” screen pops up that contains a “Play Again” button and an “Return to Main Menu” button.

S2: The Player clicks the “Play Again” button.

R2: A new instance of the game begins.

S3: The Player clicks the “Return to Main Menu” button.

R3: The frame changes back to the initial “Start Game” screen.

3.4.1.3 Functional Requirements

REQ-1: A way to know the Player interacted with the GOLDEN CONTROLLER.

REQ-2: A way to invoke the “You Win!” screen.

REQ-3: A way to know if the Player is clicking the “Play Again” button or “Return to Main Menu” button.

REQ-4: A way to start another game instance or return to the main menu.

3.4.2 Lose Screen

ID: SF 4.2

3.4.2.1 Description and Priority

When the Player runs out of brains the screen changes to the "You lose" state. It contains a "play again" button that starts a new game or an "exit to main menu button" that goes back to the initial start screen.

Priority: Medium

3.4.2.2 Stimulus/Response Sequences

S1: The Player runs out of brains.

R1: The screen changes to the "you lose" screen.

S2: The Player clicks the "play again" button.

R2: A new instance of the game begins.

S3: The Player clicks the "return to main menu button."

R3: The frame changes back to the initial "start game" screen.

3.4.2.3 Functional Requirements

REQ-1: A way to know if the Player runs out of brains.

REQ-2: A way to know if the "play again" button or "return to main menu" button is pressed.

REQ-3: A way to start another game instance or return to the main menu.

REQ-4: A way to invoke the "you lose" screen.

3.5 Menu Bar

3.5.1 Contents

ID: SF 5.1

3.5.1.1 Description and Priority

The menu is placed at the top of the frame and contains two different options, "File" and "Help." The "File" menu contains the "Save", "Load", and "Exit to start menu" options and the "Help" menu contains the "About Us" and the "How to Play" options.

Priority: High

3.5.1.2 Stimulus/Response Sequences

S1: The Player clicks the "File" menu item.

R1: A dropdown menu pops up that contains the "Save", "Load", and "Exit to Main Menu" options.

S2: The Player clicks the "Help" menu item.

R2: A dropdown menu pops up that contains the "About Us" and "How to Play" options.

3.5.1.3 Functional Requirements

REQ-1: A frame to add the toolbar to.

3.5.2 Save Game

ID: SF 5.2

3.5.2.1 Description and Priority

The toolbar has a “Save Game” menu item that allows the Player to save the current state of the game. They can specify the file name and the location the file will be saved at.

Priority: High

3.5.2.2 Stimulus/Response Sequences

S1: The Player clicks on the “Save Game” menu item.

R1: A save file dialogue box pops up that allows the Player to specify a file name and a location to save the file at.

S2: The Player clicks submit on the save file dialog box.

R2: The game is saved and stored in the file.

3.5.2.3 Functional Requirements

REQ-1: A way to save the current state of the game.

REQ-2: A way to invoke the “Save Game” dialogue box.

3.5.3 Load Game

ID: SF 5.3

3.5.3.1 Description and Priority

The toolbar has a “Load Game” menu item that allows the Player to select a game file from their computer and load that saved game’s state into the current program.

Priority: High

3.5.3.2 Stimulus/Response Sequences

S1: The Player clicks on the “Load Game” menu item.

R1: A select file dialogue box pops up that allows the Player to choose a file to load.

S2: The Player clicks submit on the select file dialog box.

R2: The game is loaded based on the given file. If the file is not valid, the game stays in its current state.

3.5.3.3 Functional Requirements

REQ-1: A way to load the game.

REQ-2: A way to invoke the “Load Game” dialogue box.

3.5.4 Return to Main Menu

ID: SF 5.4

3.5.4.1 Description and Priority

The toolbar has a "Return to Main Menu" menu item which gives the Player the ability to quit the current game. When clicked, a popup window is invoked that makes sure the Player wants to quit the current game.

Priority: High

3.5.4.2 Stimulus/Response Sequences

S1: The Player clicks the "Return to Main Menu" menu item.

R1: A popup is displayed that contains a warning about quitting the current game and a "Yes" and "No" button.

S2: The Player clicks the "Yes" button on the popup.

R2: The popup closes and the game goes to the initial title screen.

S3: The Player clicks the "No" button on the popup.

R3: The popup closes and the game continues (nothing happens).

3.5.4.3 Functional Requirements

REQ-1: A way to set up a dialogue box asking the Player if they want to quit.

REQ-2: A way to return to the main menu if the user specifies that they want to quit the main menu.

3.5.5 How to Play

ID: SF 5.5

3.5.5.1 Description and Priority

The toolbar contains a "How to Play" menu item that when clicked results in a pop-up window that displays information about how to game. The information about the game includes general information about the game, controls, and special items information.

Priority: High

3.5.5.2 Stimulus/Response Sequences

S1: The Player clicks on the "How to Play" menu item.

R1: The "How to Play" popup window appears.

3.5.5.3 Functional Requirements

REQ-1: A way to invoke the "How to Play" frame.

REQ-2: An image that displays on the frame and explains how to play the game.

3.5.6 About

ID: SF 5.6

3.5.6.1 Description and Priority

The toolbar contains an “About Us” menu item that when clicked results in a popup window that displays information about the development of the game. Information about the fame includes:

- The version
- When it was built
- What it was built for
- GitHub Repository Link

Priority: High

3.5.6.2 Stimulus/Response Sequences

S1: The Player clicks on the “About Us” menu item.

R1: The “About Us” popup window appears.

3.5.6.3 Functional Requirements

REQ-1: A way to invoke the “About Us” frame

REQ-2: An image that displays on the frame and explains the “About Us” contents.

4. External Interface Requirements

4.1 User Interfaces

The GUI standard that is used for ATM is Java's Swing library. JRadioButtons are used for the Player's options for multiple choice questions and true/false questions and JTextAreas are used for the Player's answer to free response questions. The menu bar is a JMenuBar that consists of JMenus which contains JMenuItems. The submission for the Player's answer to each respective question uses a JButton. In the future, we will extend this use of JButtons for a "Cancel" option for the trivia questions and a "Use Item" button later when items are implemented into this program.

The constraints of our program's panel layout is predetermined and static as we use fixed values for placing each component of our graphical user interface within the frame of the program.

The keyboard shortcuts for our system uses the WASD keys for the Player to move the character in the game and the Spacebar to interact with the doors in the maze and later to interact with items in the maze.

4.2 Hardware Interfaces

A monitor is required to display the graphical user interface of the program, a keyboard is needed for the Player to move around the character within the maze and answer free response questions, and a mouse is required for answering the multiple choice questions and true/false questions as well as to start the game and return to the main menu.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

A performance requirement for the amazing trivia maze requirement is that the trivia questions should be accessible and queried in a timely manner.

5.2 Safety Requirements

The Player might get the tunes in the game stuck in their head or the Player might become too addicted to our program and become a shut-in.

5.3 Security Requirements

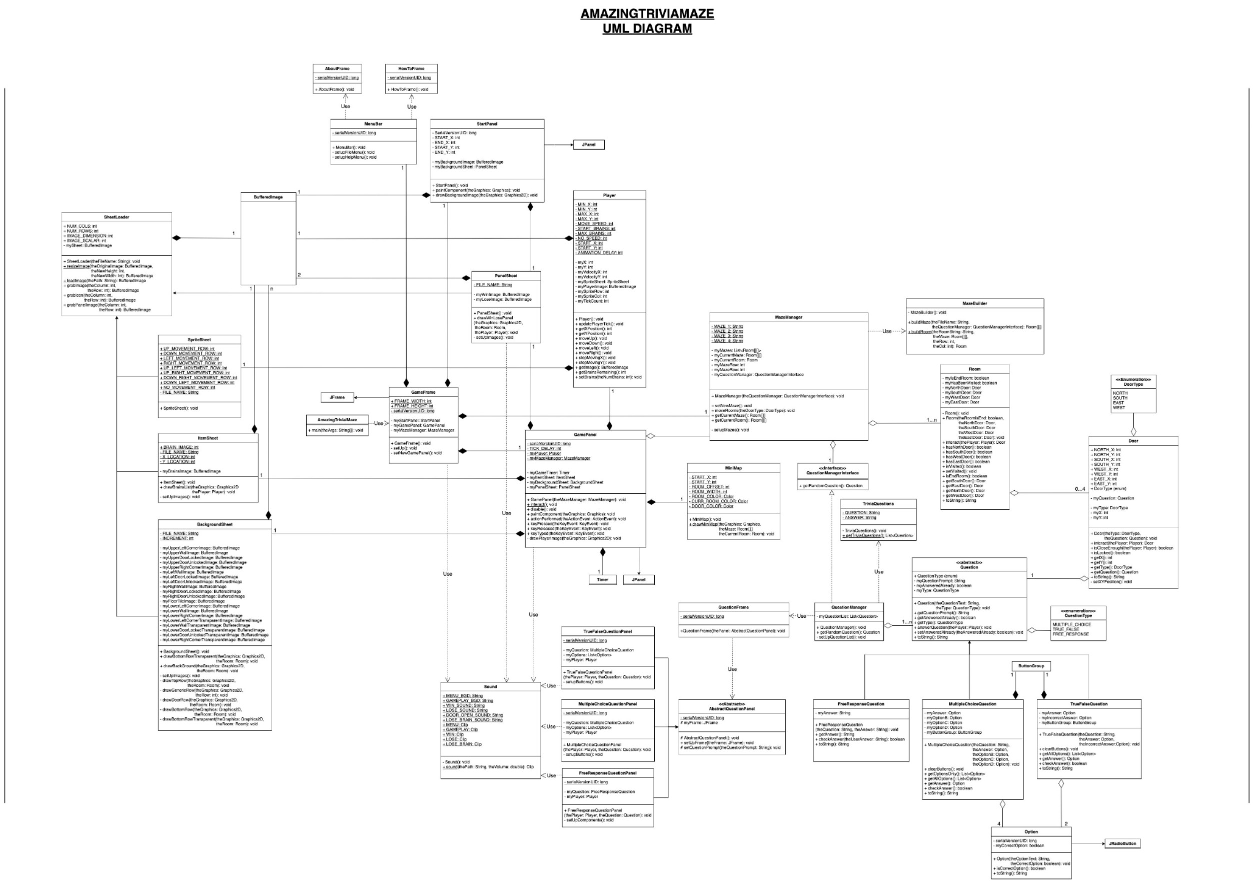
Not applicable at this time.

5.4 Software Quality Attributes

Adaptability - Able to run on multiple operating systems (at least 2).
Correctness - The maze's trivia questions are 100% accurate and correct.
Maintainability - The maze's trivia questions are easily able to be added or removed from the already 70 questions database and is easily accessible. The code base conforms to coding standards and is well organized.
Reliability - The program outputs 0 warnings or errors from the Player's interaction with the software.
Robustness - The maze builder is able to create any number of mazes based on simple text files.
Scalability - The game is able to scale the maze sizes and the number of trivia questions for the maze.
Testability - The program has nearly 200 test cases for the classes that make up the model.

6. Diagrams

6.1 UML Diagram


AMAZINGTRIVIAMAZE
UML DIAGRAM
AboutFrame
HowToFrame
MenuBar
StartPanel
JPanel
BufferedImage
SheetLoader
Player
PanelSheet
SpriteSheet
MazeManager
MazeBuilder
Room
DoorType
GameFrame
JFrame
AmazingTriviaMaze
ItemSheet
GamePanel
MiniMap
QuestionManagerInterface
Door
BackgroundSheet
TriviaQuestions
Question
Timer
QuestionFrame
QuestionManager
QuestionType
TrueFalseQuestionPanel
Sound
MultipleChoiceQuestionPanel
AbstractQuestionPanel
ButtonGroup
FreeResponseQuestionPanel
FreeResponseQuestion
MultipleChoiceQuestion
TrueFalseQuestion
Option
JRadioButton