Software Requirements Specification

for

Maze DU~~C~~KS

**Version <1.0>**

**Prepared by**

**Group Name: Sitting Ducks**

|  |  |  |
| --- | --- | --- |
| **Devdatta Mahesh Joshi** | **170953070** | [**devdattaj007@gmail.com**](mailto:devdattaj007@gmail.com) |
| **Kunal Khanwalkar** | **170953162** | [**khanwalkar@gmail.com**](mailto:khanwalkar@gmail.com) |
| **Swapnil Samanta** | **170953120** | [**swapnil319206@gmail.com**](mailto:swapnil319206@gmail.com) |
| **Utkarsh Chapekar** | **170953106** | [**Utkarshchhapekar2@gmail.com**](mailto:Utkarshchhapekar2@gmail.com) |
|  |  |  |

|  |  |
| --- | --- |
| **Instructor:** | **Rajesh Rao** |
| **Course:** | **Rational Unified Process Lab** |
| **Lab Section:** | ***B1*** |
| **Date:** | **September 16th, 2019** |
|  |  |

**Contents**

**Revisions iii**

**1** **Introduction 1**

1.1 Document Purpose 1

1.2 Product Scope 1

1.3 Intended Audience and Document Overview 1

1.4 Definitions, Acronyms and Abbreviations 2

1.5 Document Conventions 2

1.6 References and Acknowledgments 2

**2** **Overall Description 3**

2.1 Product Perspective 3

2.2 Product Functionality 3

2.3 Users and Characteristics 4

2.4 Operating Environment 4

2.5 Design and Implementation Constraints 4

2.6 User Documentation 5

**3** **Specific Requirements 6**

3.1 External Interface Requirements 6

3.2 Functional Requirements 9

**4** **Other Non-functional Requirements 10**

4.1 Performance Requirements 10

4.2 Safety and Security Requirements 10

4.3 Software Quality Attributes 10

**5** **diagrams 11**

5.1 Use Case Diagram 11

5.2 Swimlane Diagram 12

5.3 Class Diagram 13

# Introduction

## 

## Document Purpose

This Software Requirements Specification (SRS) is intended to give a complete overview of the JAVA based puzzle game Maze DU~~C~~KS, including the game mechanics, multiple game modes and functions. This document also illustrates the thought process of the creators and the inspiration behind the same. A brief overlook of the tools and methods used in realization of the game is also included.

## Product Scope

The Maze DU~~C~~KS game is a single player third person perspective puzzle game, where the primary objective of the player is to reach the end of the maze.

The Maze itself and all playable elements are made using JAVA and it’s subsidiaries.

Machine Learning is used in order to create finer aspects of the game such as adaptive difficulty.

The game can be played in various modes, each with different objectives and strategies.

An SQL-based database is used to store the player’s information and their records.

The fundamental objective of the Maze DU~~C~~KS game is to provide an entertaining experience that stimulates the player’s prefrontal cortex, to give one better problem solving skills and time based adaptiveness.

## Intended Audience and Document Overview

This SRS also entails the team’s original vision with regards to the use and proliferation of the game.

The maze game is intended for all players above the age of 5.

The game however, primarily targets the audience between the ages of 5-18.

For an overview of the Document and project itself, head to Overall Description (Section 2).

For a description of the game requirements, head to Specific Requirements (Section 3).

## Definitions, Acronyms and Abbreviations

AWT - Abstract Window Toolkit

DFS – Depth First Search

GUI – Graphical User Interface

UI – User Interface

## Document Conventions

In general this document follows the IEEE formatting requirements.

Arial font size 11, or 12 is used throughout the document for text, except for headings which are Arial font size 14.

Italics are used for comments.

The document refers to the Maze DU~~C~~KS game as “the game” or the “maze game” and any and all usage of these phrases refers to the same.

## References and Acknowledgments

We have used the following websites for inspiration for the creation of the game. NOTE: these are purely for reference purposes and the code in the same hasn’t been used.

https://www.geeksforgeeks.org/check-possible-path-2d-matrix/amp/

https://www.baeldung.com/java-solve-maze

https://dev.to/marksasp95/introducing-maze-generator-java-320g

https://softwarerecs.stackexchange.com/questions/1371/maze-generator-library

# Overall Description

## Product Perspective

This game is not a continuation of any previous product. It is a standalone application which has been developed using JAVA and JAVA FX completely.

This game uses a maze generation algorithm to generate a complete maze along with being able to generate it according to a difficulty.

The main objective of this game is for the user to navigate his pawn from the starting point in the maze to the end point in the game using arrow keys.

There are multiple game-modes to choose from like classic, arcade, time-mode and practice which will be elaborated on further.

The game allows the user to select a difficulty and it also uses adaptive difficulty which changes the difficulty of the next level based on how fast the user completes the current level.

The game is designed in such a way that the user can understand how the game works just by looking at it and without referring to a manual of some sort.

The Software provides an easy interface to play the game as well as access different aspects of the game itself.

## Product Functionality

The following is a summary of major features implemented in the game.

* Timer:

This is a basic clock that will calculate the time it takes for a user to solve a particular difficulty of the maze.

* Leaderboard:

This is a database of the ranking of all players based on the time they took to solve the maze.

* Classic Game-mode:

This is the basic game-mode of the maze. The user will have to solve the maze and there will be a timer to calculate how long the user takes to solve the maze. The user’s name will be updated on the leaderboard along with the time taken to solve the maze.

* Time attack Game-mode:

In this, the user has 5 minutes to solve as many mazes as possible.

* Arcade Game-mode:

This game mode contains different collectibles on the different paths of the maze. The different collectibles could be extra time, extra scores or other functional power ups which gives the user special advantages or disadvantages when solving the maze.

* Practice Game-mode:

This game mode has no time limit or any special feature. The user can freely navigate the maze and try to solve it. There won’t be any update of the leaderboard in this mode.

* Adaptive difficulty

The game would use Machine learning to give the difficulty of the next level the user needs to play based on the time or score the user secures in the current round.

* Hint system

In some game-modes, the user can take a hint at the cost of losing/gaining additional time. The hint would show the path the user can take to the finish from his current position, only up to some specific distance.

## Users and Characteristics

For this game there will be two main users:

**Player:** This user is the one who plays the game. He will provide the system with a username so that it can be registered on the leaderboard.

**System:** The system is responsible for the selection of the difficulty and the game-modes of the maze. It queries the database whenever a user finishes the level so that the leaderboard can be updated.

## Operating Environment

The software can work on windows 7 and above with the following hardware requirements:

* Database: SQL
* Language: Java
* Processor:  Intel dual core or above
* Processor Speed: 1.0GHZ or above
* RAM:  1 GB RAM or above
* Hard Disk:  20 GB hard disk or above

This software can also be run on Netbeans v7.

## Design and Implementation Constraints

This lists the implementation constraints the software has and the specific applications it will use for its functioning.

* The whole program is written using Java only so no other language can be implemented with it.
* MySQL is used as the database so programmers must have knowledge regarding the queries of MySQL.
* NetBeans is used as the primary interface to run this application.
* The GUI is only in English
* The parallel operations taking place in this program will be the running of the timer while the user solves the maze.
* JavaFX is the primary design tool used to design the maze.

## User Documentation

The game contains a very easy to use GUI and simple interface.

There will be one help document which shows the users the functionality of the basic GUI layout and where the user can find each function in the GUI.

For the game and the game-modes, instructions will be shown in the beginning of every game-mode on the screen just before the user begins to play the game.

# Specific Requirements

## External Interface Requirements

### User Interfaces

**Resolution**

The screen resolution constraints for the game are 640 by 480 *screen size* with each block being 40 by 40 pixels.

**Software Components**

1. The Menu: This contains the options of all the game modes the user can play in along with the option to view the leaderboard and exit the game.

1. The Leaderboard: It fetches data from the database which contains the ranks of all the

users who have played the game before, stored in the form of data such as the time taken and the no. of mazes completed in total. Any user can click on the leaderboard option in the menu to see their rank in the game so far.

1. The Game: On selecting any option on the menu, a new window opens which contains the generated maze itself.

**GUI Standards**

A SWING GUI client is used.

The GUI is built from standard SWING widgets, like

* Label
* CheckBox
* JRadioButton
* JSlider
* JComboBox
* JProgressBar
* JToggleButton
* JList
* JTextArea
* JTextPanel.

Java Awt classes are used for graphics too*.*

**Game Mode Description**

To complete any maze, the user has to find the key in the maze and go to the exit door with that key and open it.

The starting UI for the Game has 4 options for the user to choose from.

* Practice: The user can practice as much as he wants without any time limit. The user will use the arrow keys to move up, down, left and right. He cannot move through the walls. The user can select the difficulty to practice on. The leaderboard will not be affected by this game-mode.
* Arcade: The arcade mode gives the user the option of having the option of collecting artifacts along the way which provide additional advantages or disadvantages to the user which adds more features to the classic mode with a normal time limit. The user will also have to collect a key first so that he can unlock the end of the maze (which is the door).
* Time mode: The user must solve as many mazes as possible in 5 minutes. On the completion of each maze, the game engine generates a new maze of more complexity which is directly dependent on the time you had taken to complete the previous maze.
* Classic: The user solves the maze and is timed when he does so. On the completion of each maze, the game engine generates a new maze of more complexity which is directly dependent on the time the user had taken to complete the previous maze. This time is used to rank each user on the leaderboard.

The user is free to quit any game mode at any point of time. There is a cancel button located in the top right corner allowing the user to do so.

**Error Messages**

If the user tries to move when the time exceeds, A message pops up stating that the time is up.

### Hardware Interfaces

**Hardware Interactions:**

The external hardware devices being used are the mouse and the keyboard. The user uses the mouse to scroll around the homepage menu and click on buttons. The movement of the character in the maze is through the arrow keys on it which the user presses.

### Software Interfaces

The communication between the various software takes place in the form of user with the Operating system, and the game with the data base. MySQL will be used to store the usernames, the leaderboard, as well as information related to the predictive generation of the maze. As the user plays the game, the game will occasionally query the database for information.

The user logs in into his account after a login page which asks for only the username. If the username already exists, he automatically logs in, if it doesn’t, his name is automatically inserted in the database. After successfully logging in the user sees a menu of four options which opens four different interfaces.

1. Practice
2. Arcade
3. Time Mode
4. Classic

The user is free to quit any game mode at any point of time. There is a cancel button located in the top right corner allowing the user to do so.

**Libraries Used:**

This library helped us to randomly generate the maze.

1. Java FX: We used Java Swing for all the UI’s color, graphics property.
2. Java Swing: Used for the dialog boxes and overall layout of the game.
3. AWT: We also used some AWT components for buttons and labels on the UI.

### Communications Interfaces

The only communication in the game is between the system and the database. This communication happens in two cases:

1. When the user plays in the classic mode, his score is dynamically compared to the ones in the database and according to that his rank and score is updated.
2. When the user explicitly wants to click on the leaderboard option on the homepage, he can see his rank. The data is fetched from the database.

## Functional Requirements

**Maze Generation:**

Aldous-Broder algorithm is used for Maze Generation. This algorithm works exclusively for maze generation, it uses a matrix to create the paths. Made simple, this is the algorithm:

1. Take any cell randomly and check it.
2. Take any neighbor cell (not diagonal), if that cell hasn't been checked, check it.
3. Repeat step 2 until all the cells are checked.

This way the maze is generated. It will be in the form of a matrix. 8\*8,10\*10,12\*12. The higher the level, the higher the dimension of the matrix.

**Maze Solving:**

Depth First Search (DFS) algorithm is used for solving the maze.

This algorithm can be outlined as:

If we're at the wall or an already visited node, return failure

Else if we're the exit node, then return success

Else, add the node in path list and recursively travel in all four directions. If failure is returned, remove the node from the path and return failure. Path list will contain a unique path when exit is found.

**Recursive Backtracker (DFS)**

Backtracking is used along with the method of marking visited nodes to obtain a path in a reasonable time

**Leaderboard Calculation:**

The leaderboard is dynamically formed when a user is playing in classic mode. Based on the time he takes to complete each maze and the total no. of mazes that he completes, his rank in the leaderboard will be calculated. The leaderboard database will be sorted rank-wise and the person at the top of the database will the one who is first in the leaderboard. Also the user can see his rank from the main menu anytime he wants from the menu.

**Login**

The user has to enter his username to login. If his name doesn’t exist in the database, it automatically gets inserted, otherwise he just logs in normally.

**Timer**

The timer should start running as soon as a new maze is generated. This is done by the System.

**Character Movement**

This is done by the game engine based on the keyboard input it is fed from the arrow keys and the mouse. The character moves left, right, top or bottom based on the arrow key pressed. On pressing Enter, it collects the artefacts or the keys (for arcade game-mode).

# Other Non-functional Requirements

## Performance Requirements

The size of the software shall not exceed 100 Mb.

The latency between keypress and corresponding action in the environment is negligible.

## Safety and Security Requirements

The game utilizes an isolated Java Environment, there are no prospects of security breach.

The leaderboard and user data is saved in a separate local database, which does not interact with other system processes.

This product is highly secure as it does not deal with sensitive user data.

## Software Quality Attributes

### Portability

* Our primary development language is Java, which is extremely portable due to the usage of the Java Virtual Environment.
* The weights for the regression model are pre-trained and will be shipped with the final product.
* The game will be ready to use at all times without the need of any installation for OS Registry or Data setup.

### 1.2 Robustness

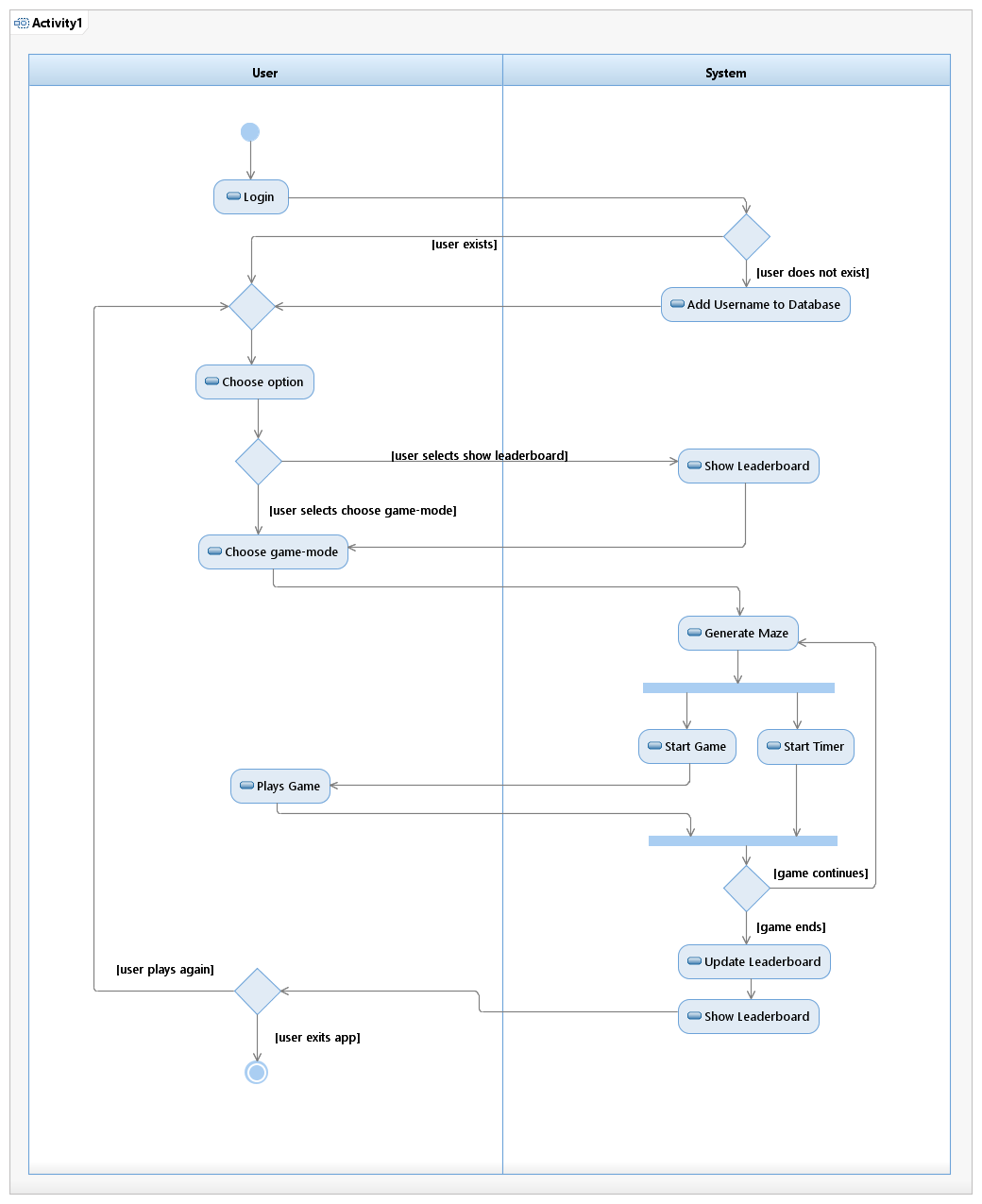
* The software is designed with the use of an incremental process model.
* Every increment handles the possibility of all errors for the added functionality at that stage.
* The entire software stack is internal without the inclusion of outer dependencies (such as internet, cloud, etc). The final product is ready to use.

### 1.3 Usability

* The UI of the product is highly intuitive with main focus on ease to use.
* A maze game aims to provide a means of entertainment for every demographic by providing a hyper-casual product for the customer to bide their time.
* The product is shipped as ‘Direct Play’.

**5.1 Use Case Diagram**

**5.2 Swimlane Diagram**



**5.3 Class Diagram**

