Software Requirements Specification

For

Labyrinth of Algorithms

Version 1.3

Prepared by Will Franzen and Lukas Livengood

Case Western Reserve University,
CSDS 393 - Software Engineering

December 9, 2022

Table of Contents

1. Introduction

1.1. Purpose

This SRS serves as a foundation for the initial demo to ensure that the implemented features meet the goals set for the application's functionality. The intended users of this document are the project members. All requirements noted in this document are to be included in the final release of the project.

1.2. Vision

1.2.1. Vision Statement

The Labyrinth of Algorithms is an educational application that provides powerful, interactable visualizations of search algorithms. Users will be able to choose from a selection of six of the most popular search algorithms and observe them as they traverse through randomized and/or user-defined obstacles. For those learning these concepts, these visualizations will help users to better understand the underlying mechanisms.

1.2.2. Major Features

FE-1:	The application will visualize a defined algorithm's traversal through grid-like mazes.
FE-2:	Users will also be able to create their own mazes.
FE-3:	Users will be able to traverse through their own mazes.
FE-4:	Solutions generated by user designated search algorithms will be visualized.

1.3. Scope

1.3.1. Scope of Initial Release

This application is a puzzle game at its core that should be able to be utilized as a tool for visualizing the step-by-step solution process of common search algorithms. The user will be able to design and traverse multiple mazes as well as follow the solutions of algorithms in their own creations, testing their knowledge of the algorithms by attempting to predict what steps it will take in its solution. The initial demo of this application should have a handful of common search algorithms implemented, such as A* and depth first search, as well as levels that we develop to test out the user traversal interface.

1.3.2. Limitations

LI-1:	Algorithms that can be visualized will be limited to ones that are implemented in our development; users will not be able to import algorithms without modifying our product.
LI-2:	Total maze size and number of tile objects will be limited by our decisions to ensure that our performance requirements are upheld on lower-end systems.

2. Overall Description

2.1. Product Perspective

Labyrinth of Algorithms is an educational platform meant to improve users' understanding of common search algorithms through features that can additionally provide entertainment value. It will provide visualizations of these algorithms in a landscape that may either be randomized or customizable by users. The application is expected to evolve over many iterations, ultimately allowing users to select from a larger range of popular search algorithms and cleaner, simpler visualizations.

2.2. User Classes and Characteristics

Learner	A learner is an end user for our system that wants to improve their understanding of search algorithms or enjoy casual use of our software. Student learners may use our platform on a weekly basis as they learn relevant material, and professional learners on a yearly basis as they prepare for job interviews.
Teacher	A teacher may choose to utilize our system to create or generate mazes for use in helping visualize the process of desired algorithms alongside their lessons.

2.3. Operating Environment

OE-1:	Our application shall be compatible with web browsers that work with the newest version of React, React 18. Target operating systems are Windows, Mac OS, and Linux devices.

2.4. Design and Implementation Constraints

CO-1:	All app functionality and classes will be written in Java.
CO-2:	All React code shall conform to the React 18 standard.

2.5. User Documentation

UD-1:	The platform will contain a help section accessible through the main area, providing basic instruction and answers to frequently asked questions.

2.6. Assumptions and Dependencies

AS-1:	The user is familiar with the concept of search algorithms.
DE-1:	At least one possible path must exist between the initial and goal state for proper functionality of the user selected search algorithm.

3. System Features

3.1. Maze Maker

3.1.1. *Create Obstacles* – The user shall be able create obstacles in the maze's individual tiles such that these tiles are to not be considered by the search algorithm.

3.1.2. *Erase Obstacles* – Existing obstacles shall also be able to be removed from the maze such that the tile they reside in is once again considered.

3.1.3. *Define Initial and Goal State* – The initial and goal state, the tiles the search algorithm will start and finish in respectively, shall be able to be defined anywhere in the maze.

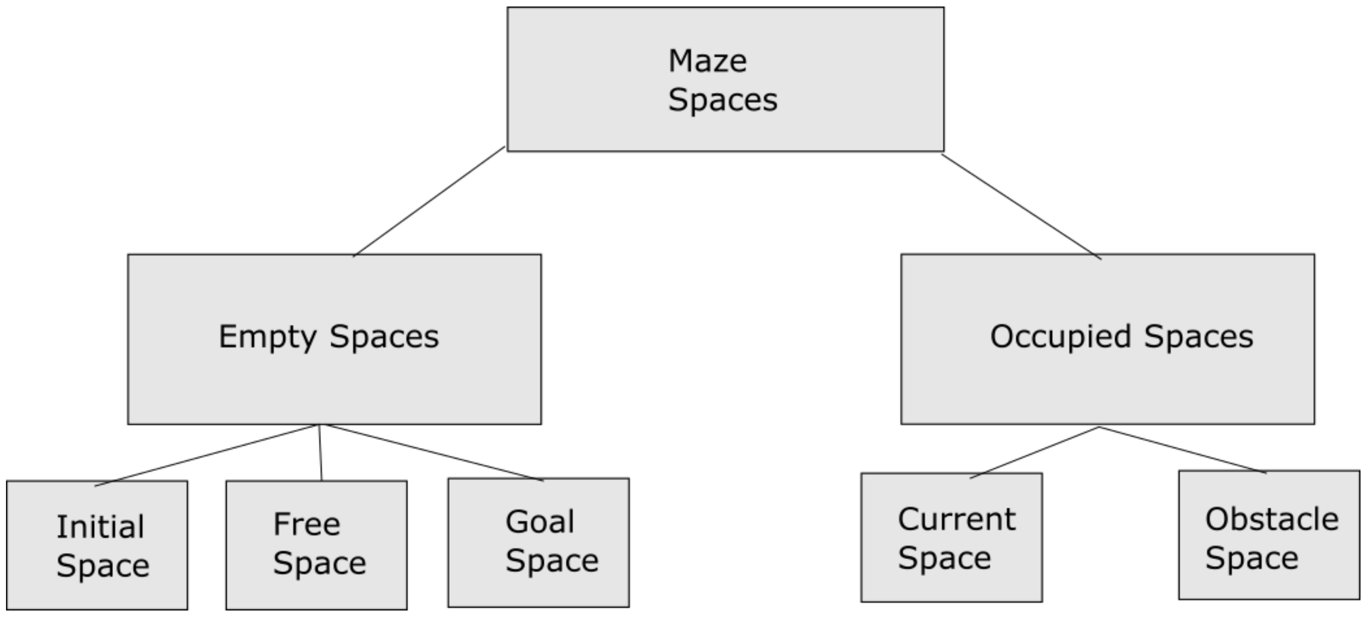

Figure 1 - Object diagram for Maze spaces. There are two types of spaces, empty spaces which can be moved into, and occupied spaces that cannot be moved into, including the space the user is currently occupying and obstacle spaces that are the walls of the maze.

3.2. Maze Solving

3.2.1. *Select Algorithm* – The user shall be able to select a search algorithm to use in the mazes traversal from a list of: Breadth-First, Depth-First, Greedy Best-First, Dijkstra's Algorithm, A* Search, Random Walk

3.2.2. *Run Algorithm* – Execute the algorithm's search and display the solution as a visualization to the user.

3.3. Maze Review

3.3.1. *Iterative Stepping* – The user shall be able to iterate through and follow the individual steps taken by the search algorithm.

3.3.2. *User Traversal* – The user shall freely be able to make adjacent movements through empty spaces in the maze between the initial and goal spaces.

4. External Interface Requirements

4.1. User Interfaces

UI-1: The main interface shall include a prominent area for the visualizations of an algorithm. The main interface shall also include an area containing features selectable by the user.

UI-2: The Maze View is the area in which the user algorithm visualization will be displayed. Additionally, it shall allow for interaction by the user such that the user will be able to place and remove walls as well as select desired locations for the initial and goal states.

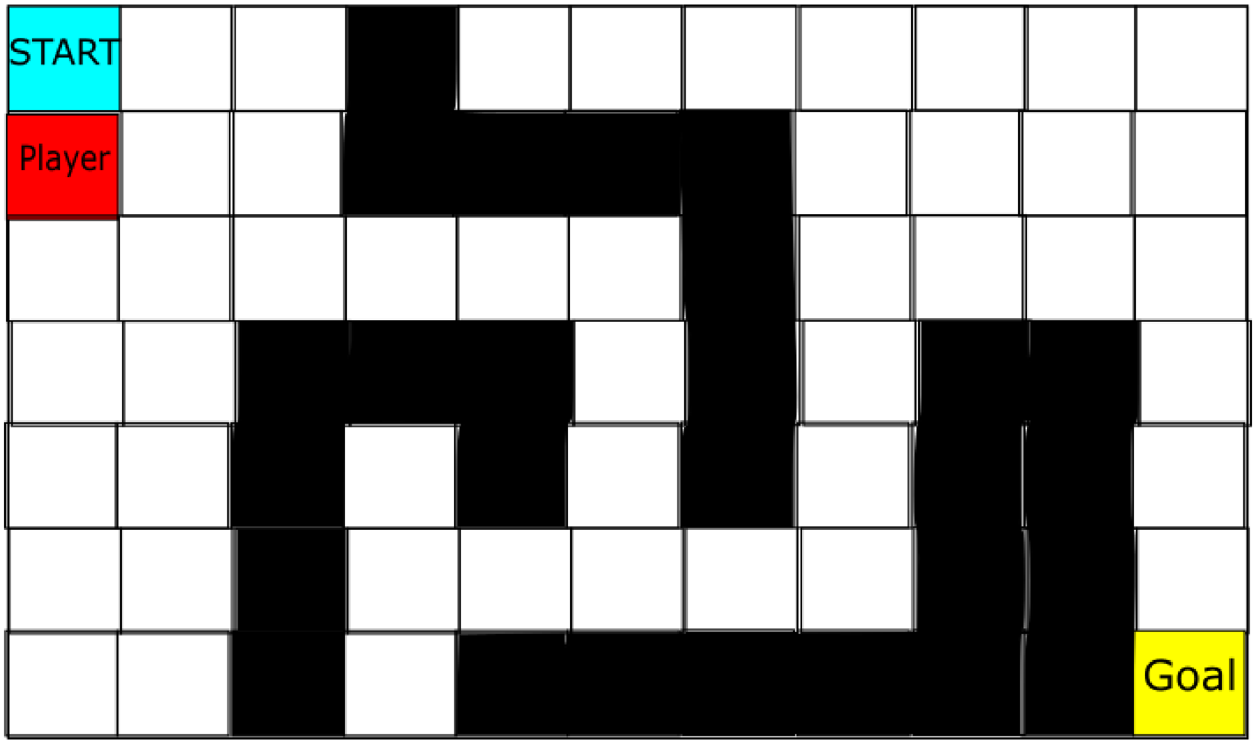

Figure 2 - A visualization of what the Maze View would look like during user traversal. Labels aren't necessary for the actual product and are present to help with identification in the image. White spaces represent traversable tiles, while the black spaces are walls that the player is not able to travel through.

4.2. Hardware Interfaces

No hardware interfaces have been identified.

4.3. Software Interfaces

No software interfaces have been identified.

5. Non-functional Requirements

5.1. Performance Requirements

PE-1:	Any maze movement actions made by the user shall occur in less than half a second after the input is registered.
PE-2:	All algorithmic moves shall occur within a second of the user attempting to move to the next action.
PE-3:	All tiles the user attempts to place shall be processed within a tenth of a second.
PE-4:	All maze creations shall be saved in less than 30 seconds upon the user's attempt to save their work

5.2. Safety Requirements

No safety requirements have been identified.

5.3. Security Requirements

No security requirements have been identified; the software does not deal with users' personal information.

5.4. Software Quality Attributes

Accurate-1:	Results produced by the user selected search algorithm should always be the first solution found and/or the most optimal solution
Robustness-1:	Obstacles, whether user or randomly created, shall not be placed in a tile occupied by the initial or goal state.

Appendix: Revision History

Date	Version	Review Outcome

9/18/22	1.0	*Initial Draft*
9/19/22	1.1	Created additional OE, CO, and UI reflecting our decision to create a web-based application
9/20/22	1.2	Final review for submission, graphical figures added where deemed appropriate.
12/5/22	1.3	Randomly generated Labyrinths have been removed. Saving and loading Labyrinths has been removed to make all functionality contained within a user session. - LL