Software Requirement Specification

1. Overview

A fundamental subject yet oftentimes challenging for software engineering and computer science students is algorithms and data structures, due to the difficulty in understanding how the algorithm works. We, Team 6, are looking to solve this problem with AlgoShadow.

AlgoShadow is an Algorithm Visualizer. With this application, users can pick algorithms, such as Dijkstra, A*, Beam Search, etc and visualize each algorithm's state as they step thru each execution step. Through this exercise, our application can help users to understand how each algorithm works.

1.1. Benefits

Here are ways our application can help users in understanding, exploring, and using algorithms in practical applications:

- Visualizing an algorithm is an effective way to enhance understanding for anyone trying to grasp complex algorithms. Visualizations can break down complex algorithms into simpler, step-by-step processes. Users can see how data is manipulated and transformed at each stage, making it easier to follow the logic
- Our application captures the actual runtime, and inputs for each experiment users ran and saves them to a database, allowing users to access their experiments later for comparing and contrasting results
- Users can use our experiment visualization feature to retrieve past data, and analyze the time complexity of each algorithm they experimented with using charts and graphs
- We provide users with a description of example real-life applications where each algorithm is applied, helping users in better understanding the algorithm's usage

1.2. Scope

This document covers requirements for both client-side app implementation and server-side implementation in release 1.0 and release 2.0. Detail of features included in each released is outlined below in User Stories section

2. High-level requirements

2.1. Functional requirements

Below is the list of functional requirement items and descriptions

Requirement item	Description
FUNCREQ1	On the Web UI, users should be able to select from a list an algorithm they want to experiment with. This list includes, but not limited to the following algorithms, Dijkstra, Bellman-Ford, Hash, Heap.
FUNCREQ2	On the Web UI, users should be able to input initial data for the algorithm they select. Then, users should be able to observe the initial state of the algorithm rendered as Graph on the UI
FUNCREQ3	Users should be able to step thru each execution step and observe algorithm's state change rendered on the UI. For example for path finding algorithm, this means as the users step through each iteration, the chosen path would be marked with a different color.
FUNCREQ4	Users should be able to reset the algorithm's state and restart the experiment

FUNCREQ4	Once an algorithm run reaches terminal state, users should be able to save the experiment results to a database. The experiment result should include actual execution time
FUNCREQ5	Users should be able to retrieve past execution results from a database. The data shall be presented on the Web UI in tabular format and in bar chart format

2.2. Non-functional requirements

Requirement item	Description
QUALREQ1	Any algorithm’s state change shall be rendered on the Web UI in less than 1 sec
QUALREQ2	Any users input must be verified, parsed, and sanitized to avoid common security risks, including but not limited to SQL Injection, cross-site scripting, and command injection
QUALREQ3	Retrieving users' past experiment data from the database to present on the UI should completes in less than 3 secs
ERRREQ1	Any exceptions that are related to running the algorithms should be displayed on the UI to allow users to react and change inputs
ERRREQ2	Any exceptions that are related to saving experiment data to a database should be logged and displayed on the UI to inform users
ERRREQ3	Any exceptions that are related to retrieving experiment data from a database should be logged and displayed on the UI to inform users

2.3. List of user stories

Below is the list of user stories:

- As users, we desire a user-friendly user interface (UI) design that simplifies interaction, providing an intuitive and pleasant experience as I engage with the learning materials and activities
- As users, we need an Algorithm page to easily access all algorithm implementations
- As users, we want a Home page for the web app to easily navigate to other page
- As users, we want to easily recognize the purpose of the webpage with relatable branding and pictures
- As users, we want a step-by-step execution capability for Dijkstra to assist me in comprehending and debugging the algorithmic logic
- As users, we want a step-by-step execution capability for BST to assist me in comprehending and debugging the algorithmic logic
- As users, we want Binary Search Tree algorithm to display the node it visited when we search for a value in the tree to be able to understand it's logic
- **A**s users, we want a step-by-step execution capability for Hash to assist us in comprehending and debugging the algorithmic logic
- As users, we want a step-by-step execution capability for Heap to assist us in comprehending and debugging the algorithmic logic
- As users, we need Create ability in Heap, and BST to allow us to create new Tree from a comma separated inputs, such as 1,2,3,4
- As users, we need Insert ability in Heap, and BST to allow us to insert a new value to the tree from an input, such as 1
- As users, we need Delete ability in Heap, and BST to allow us to delete existing value from the tree from an input, such as 1
- As users, we need Search ability in Heap, and BST to allow us to search for an existing value from the tree from an input, such as 1

- As users, we need Create ability in Dijkstra to allow us to create a new graph from an input
- As users, we need to access our data securely with proper authentication
- As users, we want to be able to save experiment data to a Database to retrieve later
- As users, we need an Experiment Table for displaying total run time, input, and output shortest path for Dijkstra
- As Users, we need a the same Experiment Table for displaying total run time, input, and output for BST, Heap, and Hash
- As users, I need a "Operation" field in the Experiment Table to distinguish run time for Create, Search, Insert, and Delete operations for BST, Heap, and Hash
- As users, we want algorithm page to have the same navigation bar to navigate to other webpage
- As users, I need helper messages in Algorithm page next to each input box and button for clearer instructions
- As developers, we want tests implemented and automated so we are more confident when merging our code

2.4. Use cases

2.4.1. Pick an algorithm

Main success scenario

1. From the Home page, user selects Algorithm page
2. User selects an algorithm name from a drop down list
3. A blank canvas is rendered
4. User provides input data to instantiate the algorithm
5. A new blank graph is rendered within the canvas
6. User clicks **Start**
7. A new button **Next** appears
8. User clicks Next
9. The algorithm moves on to its next state
10. The algorithm's new state is rendered as a graph. Processed inputs are then marked in a different color than unprocessed one
11. The algorithm reaches the terminal state
12. The algorithm final state is rendered as graph
13. A new button **Save** appears
14. A summary of the experiment appears that shows total execution time
15. User clicks **Save**
16. The experiment data is saved to local storage

2.4.2. Retrieve user data

Main success scenario

1. From the Home page, user selects Experiments page
2. A loading icon appears, which indicates user data is being retrieved from server
3. User's experiment data loaded successfully
4. User's experiment data is rendered as a table
5. User's experiment data is rendered as a bar chart

2.5. High-level UI design

Below is the current design for Algorithm page. Users will be able to select algorithm they want to experiment with. Next, the algorithm graph will appear on within the grey canvas. Users can then use the **Start** button to start stepping through the algorithm and observe its state changes. A **Reset** button allows the users to clear all states and restart.

In addition to the **Algorithm** page, **About** page has a short descriptions of the project. Finally, the **Contact** page has emails and contact information of the team.

For most updated UI Design, please also visit https://www.figma.com/file/pULQC1OwkhB9GCFttLYN7d/Algo-Shadow?type=design&node-id=0-1&mode=design&t=8QuIa69w4l0Y8FAH-0