MET CS 673 Project Proposal: AlgoShadow

Product name Algo Shadow proudly sponsored by OpenAI ChatGPT. We also used ChatGPT to validate grammar.

Team 6 point of contacts

- Product Manager: @My Nguyen
- UI/UX Designer and Tester: @Fengyun Chen
- Front-end Developers: @Yutong Feng @Junyi Ma @Xinyue Chen
- Back-end Developer: @Xinyue Chen
- Documents maintained by: @My Nguyen

1. Product Description

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.

2. 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

3. Target users

Our application is intended for students or professionals, who want to learn or refresh their knowledge of Algorithms and Data Structures.

4. Core features

This application would have 2 main components, a front-end application and a back-end server.

4.1. Front-end application

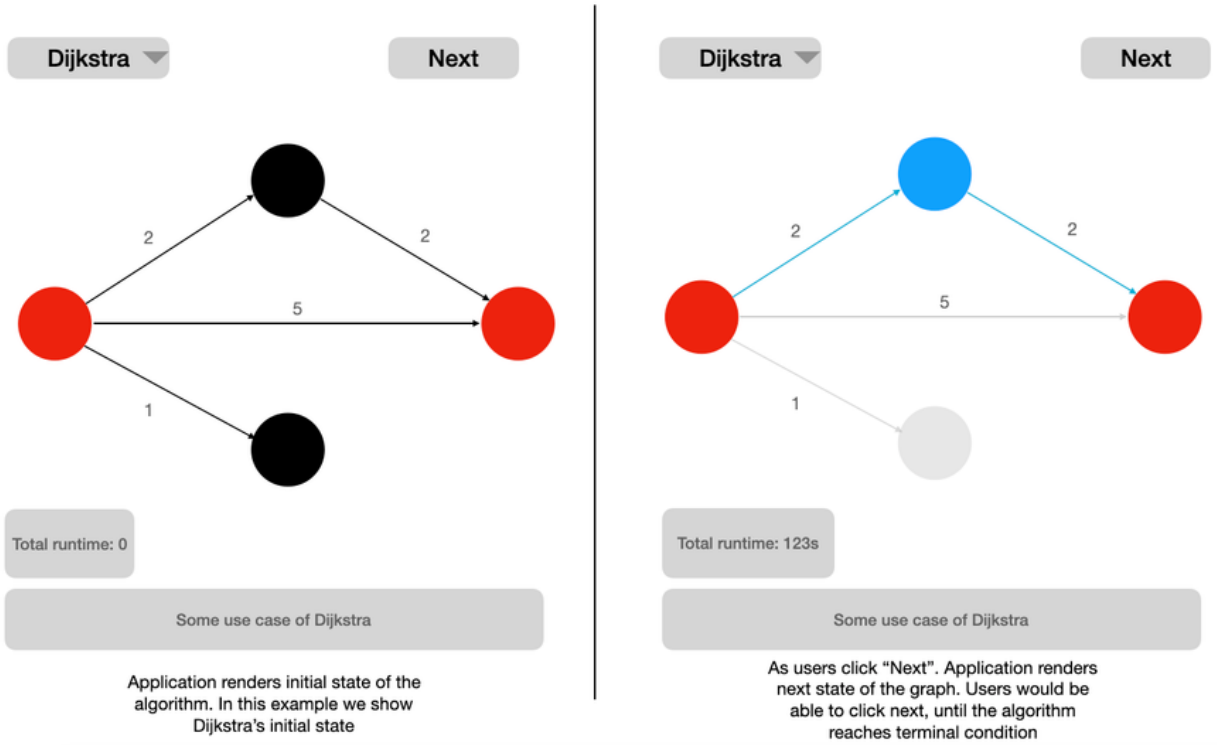
The front-end application includes a web user interface and an algorithm visualizer component written in React.js that is run and rendered on client-side.

4.1.1. Algorithm visualizer view

Users should be able to do the following in the user interface:

- Pick an algorithm listed in the application
- Start an algorithm experiment by instantiating the algorithm's state and inputs
- Step through each execution step. In each step, as algorithm state and data structure change, a graph or animation should also change to reflect the new state

Below visual shows the early concept we have for this application



For 1.0 release, we include Heap, Hash, Bellman-Ford, Dijkstra, and Binary Search Tree algorithms. Additional algorithms will be added in subsequent releases

4.1.2 Experiment view

After completing each experiment, users can chose to send following data points to the back-end server to be stored to a database:

- algorithm selected
- algorithm inputs
- actual execution time

Then, users can retrieve past data and view it using our built-in charts and graphs in "Experiment view".

4.1.3 Client-side authentication

Users are presented an option to login if they want to save their experiment data to retrieve later.

4.2. Back-end server and database

4.2.3 Server-side authentication

Server-side authentication validates token sent by client-side application for authorized access. Authorized users are able to send and retrieve experiment data from the server.

4.2.1 Collect and return experiment data

Through a REST API, client-side application can get and retrieve data for a particular user from a database.

5. Technology Stack

- For the front-end application, we are using React.js to implement the algorithm component and Material UI to implement UI. Charts and graphs capability in Experiment View is implemented using D3.js
- For the back-end server, we are using Java SpringBoot framework to implement Rest APIs and server logic
- For the database we are using PostgreSQL to store and manage user data

6. Software Configuration Management Plan Summary

Below is a summary of tools and platforms we are using to store documentations and code files, with links:

Code files version control repository	Github
Project documentation (meeting minutes, project status, design docs, and development artifacts)	Confluence
User Manual and Demo	Confluence
Developer instructions	Github
Project management (stories, tasks, delivery timeline)	Jira
Meeting and communication platform	Discord
IDE	Visual Studio Code / WebStorm

7. Project Timeline

Bellow is the current project timeline (subjected to change):