sddec23-14

Interactive Evaluation of Shortest Path Methods

Design Document

Client: Goce Trajcevski

Advisor: Mengxuan Zhang

Alex Blomquist	Backend Development / Design Integration
Samuel Caldwell	Frontend Development/ Component Design
Selma Saric	Frontend Development/ Team Organization
Yadiel Johnson	Backend Development / Documentation Management

Website: http://sddec23-14.sd.ece.iastate.edu/

Contact: sddec23-14@iastate.edu

Executive Summary

Development Standards & Practices Used

- IEEE 26514-2021 – System and Software Engineering
- IEEE 29119-1-2021 – Software and Systems Engineering
- IEEE 42010-2022 – International Standard for Software, Systems, and Enterprise
- Scrum Methodology

Summary of Requirements

Functional/Specification:

- Enable seamless integration of multiple algorithms and their execution of various datasets.
- Provide correct measurements of the runtime of different algorithms on individual datasets.
- Distinguish algorithms based on different edge weights (e.g., traffic data changes)
- Allow users to select datasets and algorithms to test.
- Provide informative visualizations of algorithm outcomes and compare these against each other.
- Generate a report of algorithm efficiency and related data.
- Provide the user with the format to create datasets for executing algorithms, such that the product can use them.

Resource Requirements:

- Use resources optimally per algorithm run.
- Support parallel, independent workloads.
- Store reports and records of algorithm-dataset runs.

UI Requirements:

- Provide a method to visualize final executions of the algorithms.
- Display the shortest path between the given source and destination as advised by a given algorithm.
- Allow users to upload datasets and select which algorithms to test them against.
- Present information neatly for easy understanding.
- Provide detailed information as an option for reports and visualizations.

Constraints:

- The system must provide all functionality as a full-stack solution.
- The overall implementation should be within a “reasonable” budget (e.g., no more than $200).

Applicable Courses from Iowa State University Curriculum

- COM S 228 – Introduction to Data Structures
- COM S 309 – Software Development Practices
- COM S 311 – Introduction to the Design and Analysis of Algorithms
- COM S 319 – Construction of User Interfaces
- COM S 329 – Software Project Management
- COM S 363 – Introduction to Database Management Systems
- CPR E 416 – Software Evolution and Maintenance
- S E 317 – Introduction to Software Testing
- S E 339 – Software Architecture and Design

New Skills/Knowledge acquired that was not taught in courses.

- Scalability of shortest path algorithms for large datasets
- MapBox API
- Graphology/Gram

Table of Contents

List of Figures

List of Tables

1 Team

An introduction to the team and their working dynamic can be found in this section.

1.1 Team Members

The members of team sddec23-14 (henceforth referred to as "the team") can be found below, alongside their self-reported proficiencies, skill sets, and undergraduate major.

Name	Proficiencies	
Alex Blomquist Computer Engineering	Full-stack Development (preference: Backend)	Git, CI/CD Java, C, SQL, HTML Linux/Unix, Bash
Samuel Caldwell Software Engineering	Full-stack Development (preference: Frontend)	Git Java, Kotlin, C, C++ JavaScript
Selma Saric Software Engineering	Project Management	Project/product management experience Good scheduling, coordination, communication, organization skills
	Frontend Development	Git Java, C HTML, CSS, and JavaScript
Yadiel Johnson Software Engineering	Server Management	Linux/Unix, Bash, networking, server maintenance & provisioning
	Backend Development	Git, Docker, CI/CD, website deployment Java, C, C++, SQL

Table 1.1 - Team Members and Skill Sets

1.2 Project Management Style

The team elected to utilize the Agile methodology; it was found to be well-suited for the project as it emphasizes collaboration, flexibility, and constant feedback. Later sections, such as section 3.1, demonstrate how the flexibility of Agile allows the team to address feedback while guaranteeing high quality software through peer reviewing methods and continuous improvements. The team also elected to use Trello, which helps keep every team member up to date on project tasks that are finished or have yet to be done.

1.3 Required Skill Sets

Skill sets required to properly design and develop the project are listed below.

Skills	Requirement
Frontend Development	The project must feature an interactive component where users can submit data sets and visualize graphs. Ordinary webpage aspects, such as form submission, must be complemented with visuals, asynchronous responses, and dynamic breakdowns of each run.
Backend Development	The project must store and process algorithms and data sets for use on command. This includes managing variable workloads, focus on reliability, and state management. Incorporating databases and other persistence methods will be necessary.
Project Management	The project must be managed in an organized and efficient way throughout the project's lifetime. This is necessary to ensure that the team can plan to finish tasks on time and figure out exactly what needs to be worked on next.
Domain Knowledge	The project must be a capable educational tool to represent an algorithm's strengths and weaknesses. Understanding what users may seek from this tool is an important aspect in its development, and likewise the implementation of these algorithms benefits from domain knowledge.

Table 1.2 - Required Skill Sets

1.4 Project Management Roles

The roles each team member will fulfill throughout the design and development process can be found below.

Name	Initial Roles
Alex Blomquist	Setup & coordination of design documentation Backend Development, Continuous integration
Samuel Caldwell	Component Design/Architecture Frontend Development
Selma Saric	Project Manager Meeting Coordinator Meeting Minutes Recorder Frontend Development
Yadiel Johnson	Documentation Manager Backend Development, CI/CD

Table 1.3 - Project Management Roles

2 Introduction

Algorithm research has come up with different variations for shortest-path calculations that have their own limitations, proficiencies, and intended use cases. Efficiency is always a major consideration for them and it is often complicated to explain just how efficient one is relative to the other. Because there is not a clear way to compare or visualize these algorithms, it impedes researchers from explaining their work, educators from teaching students, and the general audience from understanding the purpose of their existence. The nature of this problem is twofold:

1. It can prove difficult to demonstrate your findings without practical examples or comparisons.
2. How do you allow various people to verify these claims at any time, regardless of knowledge-level?

Our project aims at addressing these needs by developing a proof-of-concept implementation of a system with various algorithms and datasets available for testing, and outputting detailed comparisons and reports among them. This kind of comparative knowledge is of great importance to researchers, engineers, and designers alike whose focus is on understanding and implementing the optimal algorithms for a given context.

2.1 Engineering Standards

The following is a listing of the various engineering standards that the team will adhere to, with the goal of ensuring that both the design and development process is consistent and high quality. They have been meticulously selected to align the project's development lifecycle to the industry's best practices, and by following their guidelines, the team agrees that they will strengthen both aspects of this proof-of-concept.

ISO/IEC/IEEE 26514-2021- Systems and Software Engineering

Design and Development of Information for Users [1]

> To provide the user with the tools to make the most informed decision when picking an algorithm, it's important to keep everyone on the team informed through familiar and standardized documentation. The standard above provides a guide to a consistent approach to document creation and management throughout the project. Within the framework of our project, this will assist us in the design documentation of the graphical user interface, API, and many areas throughout the software system in a manner that provides structure and consistent formatting of information for all project stakeholders.

ISO/IEC/IEEE 29119-1-2021 - Software and Systems Engineering

Software Testing -- Part 1: General Concepts [2]

> This document provides guidance on creating a comprehensive and consistent approach to software testing. Additionally, it provides insights to reduce risks associated with software failures with the aim of improving the quality of a software product. In the context of our project, this will guide us in designing the respective unit, integration, and acceptance testing scenarios, as well as in defining meaningful regression tests.

IEEE/ISO/IEC 42010-2022 - International Standard for Software, Systems and Enterprise

Architecture Description [3]

> The document emphasizes focusing on stakeholder perspectives while creating the architecture and provides guidance for documenting the design process. Because the project is aimed at researchers and educators, who have very particular needs and certain expectations regarding the system's features, it is important to accurately identify their perspectives and concerns while also designing a product that thoroughly addresses them.

Further references to the engineering standards shall follow the format:

```
IEEE <###> section <###>
```

Where the revision year is implied to be the year associated with the standards above.

2.2 Stakeholders

In accordance to IEEE 42010 section 6.2 and 6.3, the following stakeholders were identified for this project, alongside the perspectives they may hold regarding it:

SH-1 Researchers

- Users who will use the product to generate reports on the efficacy of various shortest-path algorithms, including comparisons between them. They will use the product to choose which algorithms are ideal for their own projects based on the peculiarity of the datasets they work with.

SH-2 Educators

- Users who will use the product as a tool for education, teaching students about the various shortest-path algorithms. They will use the product to demonstrate the purpose of each algorithm and how they arrive at their conclusions.

SH-3 Students & Casual Users

- Users who will use the product for nondescript purposes, such as deepening their knowledge about shortest-path algorithms.

SH-4 Application Maintainer

- The person, or team, who will be responsible for the product's upkeep and maintenance over time, ensuring that it can continue to meet other stakeholder's expectations.

2.2.1 Expectations and Concerns

In accordance with IEEE 42010 section 6.4, a definition of the expectations and concerns identified can be found in Table 2.1 - Stakeholder Expectations and Concerns

below, alongside their associated stakeholders.

#	Description	Stakeholders
EX-1	The product will provide complete and accurate results for a given algorithm & dataset combination.	SH-1, SH-2, SH-3
EX-2	Comprehensive documentation detailing the implementation of each algorithm, including any unique aspects, will be provided.	SH-1, SH-2
EX-3	The product allows for customized datasets tailored to the data and domain relevant to the user's field.	SH-1
EX-4	The product features a selection of readily available datasets to test.	SH-2, SH-3
EX-5	The product can provide detailed comparative metrics about multiple algorithms executed on compatible datasets.	SH-1, SH-2, SH-3
EX-6	The product will be usable on traditional desktop consumer devices with minimal compromises.	SH-1, SH-2, SH-3, SH-4
EX-7	The product includes graphical representations of algorithm executions that are clear and easy to understand.	SH-1, SH-2, SH-3
EX-8	The graphical user interface is easy to navigate and allows for intuitive operation of the product.	SH-1, SH-2, SH-3
EX-9	The product utilizes a reasonable amount of system resources per algorithm execution.	SH-4
EX-10	The product is extensible with future algorithms, datasets, and functionality.	SH-4

Table 2.1 - Stakeholder Expectations and Concerns

Additionally, a use case diagram that demonstrates the interactions between these stakeholders and the system is included below in Figure 2.1.

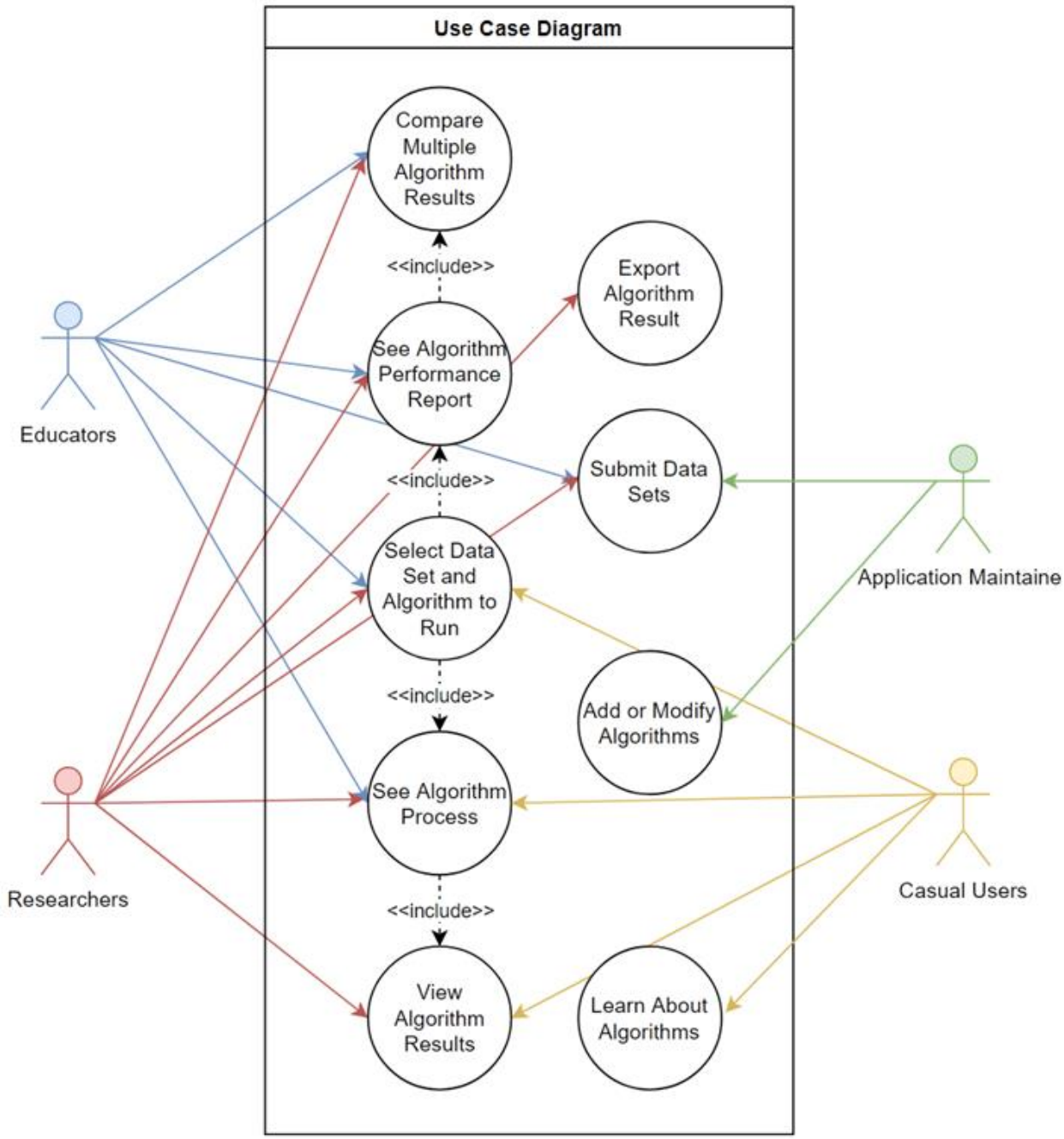


Figure 2.1 - Use Case Diagram

2.3 Requirements and Constraints

The requirements and constraints for this project are derived from the stakeholder expectations defined in section 2.2.1 and developed with the client's advisory[1].

2.3.1 Functional Requirements

To fulfill the purpose of the project and address the problem statement, the product must:

- Enable seamless integration of multiple algorithms and their execution of various datasets.
- Provide correct measurements of the runtime of different algorithms on the individual datasets.
- Distinguish algorithms based on different edge weights (e.g., traffic data changes)
- Allow users to select datasets and algorithms to test.
- Provide informative visualizations of algorithm outcomes and compare these against each other.
- Generate a report of algorithm efficiency and related data.
- Provide the user with the format to create datasets for executing algorithms, such that the product can use them.

2.3.2 Resource requirements

Given that we aim for a proof-of-concept implementation, immediate requirements are generally lax; a simple server will suffice. That said, the implementation must:

- Use resources optimally per algorithm run.
- Support parallel, independent workloads.
- Store reports and records of algorithm-dataset runs.

2.3.3 UI requirements

The user interface must:

- Provide a method to visualize final executions of the algorithms.
- Display the shortest path between the given source and destination as advised by a given algorithm.
- Allow users to upload datasets and select which algorithms to test them against.

In addition, there are qualitative & aesthetics requirements that should be met.

- Present information neatly for easy understanding.
- Provide detailed information as an option for reports and visualizations.

---

[1] Note that supplemental graphics are available in

Appendices.

2.3.4 Constraints

The constraints defined for this project are minimal but significant, nonetheless.

- The system must provide all functionality as a full-stack solution.
- The overall implementation should be within a "reasonable" budget (e.g., no more than $200).

3 Project Plan

As you will see later on, the project plan consists of creating a web application with various algorithms and datasets available for testing, as well as outputting detailed comparisons and reports between them.

3.1 Project Management & Tracking Procedures

The chosen methodology for this project was Agile due to some advantages that it has over the Waterfall method. Namely, the flexibility to adjust our priorities throughout the course of the project would fit better with the Senior Design course layout, and it allows for the client to have a higher influence throughout the project's development. The agile methodology provides various tools to enable this via providing a clearer understanding of the progress of the project as well as keeping team members updated on each other's work to identify potential roadblocks or issues during development.

The Agile procedures in use can be found below:

- Every Tuesday involves a breakdown of current and upcoming assignments and goals. These are broken down into tasks that are submitted to our team's Trello board.
- The team maintains communication via Discord, where general discussion takes place. Team meetings are also held over voice channels.
- The team has access to a GitLab instance for the code's version control. It is reserved for the implementation phase of the project.
- Retrospectives after each sprint where the team and its members can reflect on their performance and identify areas of potential improvement.

3.2 Task Decomposition

The project's task decomposition is presented in two ways; a project conceptualization and a tabular rundown of the decomposition.

3.2.1 Project Conceptualization

The team's derivation of what parts of the project need to be implemented to achieve the desired outcome can be found in this section. The approach settled on is reflective of a Model-View-Controller

design that leverages RESTful logic, and a further specification on the server side regarding an algorithm execution driver.

Frontend

This section contains the preliminary design of the client-side components.

- Create wireframes for the entirety of the web application to conceptualize its user interface.
- Create UI for the web app using HTML, CSS, and JavaScript
    - Develop a way for users to upload data sets.
    - Develop a way for the users to select algorithm(s) to run on their data sets.
    - Develop shortest-path algorithm visualizations.
    - Present algorithm runtime and metrics on the results screen.
    - Develop a method to generate reports that have comparisons between algorithms, including a method to store them.

Backend

This section contains the preliminary design of the server-side components.

- Obtain and adapt implementations of the various shortest path algorithms as described in our Taxonomy document.
    - Modify algorithm implementations such that all I/O operations are standardized.
- Develop a "driver" that receives a dataset and a requested algorithm to run against it.
- Develop a server component that manages transactions with the web application and the algorithm driver.
    - Develop the "Controller" portion of the MVC pattern to communicate with the web application via a RESTful API.
    - Integrate with the driver to coordinate multiple algorithm executions.
    - Implement methods to receive, validate, and manage datasets submitted by users.

3.2.2 Tabular Rundown

For conciseness, Table 3.1 shows all the major tasks that are necessary for the successful completion of this project, along with the corresponding sub-tasks.

Note that the corresponding sprints that illustrate the Agile methodology can be found in section 1.2.

#	Description
1	Design the System Architecture
1.1	Design the server component
1.2	Design the driver component
1.3	Design the web app component
1.4	Adjust and adapt the algorithm-dataset suite
2	Design the System Framework
2.1	Design a standardized format for algorithm I/O
2.2	Design the REST endpoints
3	Design the Testing Framework
4	Finalize Design Document
5	Prepare Server Environment
6	Implement the Server Component
6.1	Add REST endpoints
6.2	Implement persistence
7	Implement the Driver Component
7.1	Implement algorithm interface solution
7.2	Add “runtime and space complexity” metric gathering
8	Implement the Web App Component
8.1	Implement basic UI
8.2	Add REST logic
8.3	Add user form submission
8.4	Implement algorithm output visualization
8.5	Add “comparison export” functionality
9	Implement Testing Suite
10	Final Presentation

Table 3.1 - Task Decomposition Rundown

3.3 Project Milestones, Metrics, and Evaluation Criteria

A detailed listing of the milestones for this project, and their proposed deadlines, can be found below.

#	Milestone	Description	Date
1	Finalize System Architecture Design	Finalize a design concept that showcases how the frontend, driver, and backend of the system work together.	April 2nd
2	Finalize Design Document	Finalize the design document to work as an exhaustive summary of the details of the software's development.	April 23rd
3	Acquire and Adapt Algorithm Code	Receive and analyze code provided by the project advisor in order to utilize it with the AED.	Sept. 10th
4	Develop the User Interface	Integrate the initial design of the UI into the frontend of the system architecture.	Nov. 11th
5	Develop Algorithm Execution Driver	Develop a module tasked with managing algorithm executions, runtime metrics, and related responsibilities.	Oct. 1st
6	Develop the Server Component	Implement the environment and server where all web application requests will be handled, including algorithm executions and persistence.	Sept. 17th
7	Unit Testing	Ensure the functionality of each component by testing its functionality for potential software bugs during their operation.	Oct. 17th
8	Implement Algorithm Visualization	Include visualizations of algorithm results and traversal paths on the web app results screen.	Nov. 1st
9	Integration and Acceptance Testing	Test each component for their compatibility during interactions to detect any potential bugs or other potential system vulnerabilities during integration.	Nov. 17th
10	Final Software Release	Prepare the finalized version of the software system for presentation, with added emphasis on quality assessment.	Dec. 3rd
11	Final Presentation to Panel	Present the finalized version of our presentation with a demonstration of the software developed that showcases its functionality.	Dec. 8th

Table 3.2 - Milestones and Proposed Deadlines

3.4 Project Timeline

The project timeline is presented below, divided into two semesters for readability[2].

3.4.1 Project Schedule, Semester 1

Ordinarily, the first semester of a Senior Design project is a mixture of design documentation and project development. At the request of the client, however, the team has focused entirely on the documentation in order to fully develop the proof-of-concept's design.

The following figure is a breakdown of the design process for the first semester.

Task	Start	End
Phase 1: Research and Planning		
Discover Phase | Research	2/14/23	2/14/23
Phase 2: Documentation		
Team Initiation Assignment	2/14/23	2/19/23
Professionalism Assignment	2/20/23	2/26/23
Requirements, Constraints, and Engineering Standards	2/27/23	3/5/23
Senior Design Team Website, version 1	3/6/23	3/12/23
Project Plan Assignment	3/13/23	3/26/23
Design Assignment	3/27/23	4/2/23
Testing Assignment	4/3/23	4/9/23
Senior Design Team Website, version 2	4/10/23	4/23/23
Phase 3: Refinement and Presentation		
Final Design Document	4/10/23	4/23/23
Faculty Panel Presentation	5/3/23	5/3/23

Table 3.3 - Project Schedule, Semester 1

[2] Note that supplemental graphics are available in 9.1 Project Schedule Gantt Charts

3.4.2 Project Schedule, Semester 2

Likewise, Table 3.4 is a breakdown of the development process for the second semester, separated into sprints.

Task	Assigned To	Start	End
Sprint 1: Forming Frontend and Backend			
Wireframe Web App Pages	Frontend Team	8/24/23	9/3/23
Create Home Page	Frontend Team	9/4/23	9/17/23
Develop Algorithm Selection	Frontend Team	9/4/23	9/10/23
Create Ability to Upload Data Set	Frontend Team	9/11/23	9/17/23
Develop Server Controller & Persistence	Backend Team	8/24/23	9/10/23
Develop Server REST Logic	Backend Team	9/11/23	9/17/23
Unit Testing	Both Teams	9/18/23	9/30/23
Sprint 2: Algorithm Implementation and Visualization			
Develop Algorithm Visualization	Frontend Team	10/1/23	10/13/23
Implement Web App REST Logic	Frontend Team	10/14/23	10/16/23
Aggregate Algorithm Implementations	Backend Team	10/1/23	10/13/23
Develop Algorithm Execution Driver	Backend Team	10/1/23	10/16/23
Unit Testing	Both Teams	10/17/23	10/31/23
Sprint 3: Establishing Communication Between Frontend and Backend			
Connect Algorithms to Visualizer	Both Teams	11/1/23	11/6/23
Display Algorithm Runtime	Frontend Team	11/7/23	11/11/23
Create Report Generation and Storage	Both Teams	11/12/23	11/17/23
Unit Testing	Both Teams	11/17/23	12/3/23
Sprint 4: Wrapping Up			
Final Presentation to Panel		12/4/23	12/8/23

Table 3.4 - Project Schedule, Semester 2

3.5 Risks, Management & Mitigation

Table 3.5 outlines the potential risks associated with the project, the strategies for their management, and the mitigation plans to minimize their impact, as guided by IEEE 29119 section 4.2.2.

#	Title	% Risk	Reason	Mitigation Strategy
1	Design System Architecture	0.3	Planning Stage	Strict adherence to the engineering standards proposed, specifically IEEE 42010, "Architecture Description".
2	Design System Framework	0.3	Planning Stage	N/A
3	Design Testing Framework	0.2	Initialization Stage	Strict adherence to the engineering standards proposed, specifically IEEE, "Software Testing".
4	Prepare Server Environment	0.2	Initialization Stage	N/A
5	Finalize Design Document	0.3	Documentation	N/A
6	Implement Server Component	0.2	Implementation Failure	N/A
7	Implement Driver Component	0.5	Implementation Failure	Test algorithm implementation to make sure they produce the expected results.
8	Implement Web App Component	0.4	Implementation Failure	N/A
9	Implement Testing	0.3	Testing Failure	N/A
10	Final Presentation	0.4	Documentation	N/A

Table 3.5 - Risks, Management, and Mitigation Strategies

3.6 Personnel Effort Requirements

Table 3.6 outlines the human resources necessary for successfully completing the project; it is important to note that these hours are estimations based on expected difficulty of each task.

#	Title	Hours	Explanation
1	Design System Architecture	30	The server, driver, and web app components will provide the foundations for the application to work.
2	Design System Framework	30	Design the applications functions which will interact with the

3	Design Testing Framework	40	Create a test suite for verifying the application works as intended.
4	Prepare Server Environment	40	Obtain a server and prepare it to be able to store user information and algorithms.
5	Finalize Design Document	20	Complete documentation describing the application's design in its entirety.
6	Implement Server Component	50	Implement a backend housing the algorithms utilizing a Java backend to interface with the driver component.
7	Implement Driver Component	50	Implement the algorithm functions and ensure they produce the expected results.
8	Implement Web App Component	50	This will involve implementing the UI and user functions to allow them to utilize the algorithms and data sets. It will also need to retrieve the metric gatherings and present them back to the user.
9	Implement Testing	40	Final round of testing all functionalities of the application.
10	Final Presentation	20	It will be necessary to develop a concluding presentation that will highlight our project's functionality along with its associated design methodology.

Table 3.6 - Expected Personnel Effort

3.7 Other Resource Requirements

In unison with the client, only two other extraneous requirements were identified:

- Throughout the second semester, the team may request a server from Iowa State University's Electronics and Technology Group to host the backend portion of the application.
- The project is primarily focused on the proof-of-concept aspect; no financial constraints were identified by the client or the team.

4 Design

This section presents the design considerations and processes undertaken to develop a visualization tool for algorithm comparison. Given the variety of the stakeholders and their needs (seen in section 2.2.1), the design exploration was comprehensive in order to provide a flexible system.

4.1 Design Context

An examination of the context behind the proposed design can be found below.

4.1.1 Broader Context

The project's design is contingent on showing a successful application of an algorithm visualization and metric measurement tool in a research and higher-education setting. In particular, it seeks to aid researchers interested in optimizing road networks and shortest path algorithm selection. As a result, the project does not have a major stake in many of these areas (e.g., the project has little influence on the environment by itself). Therefore, the following section describes the effects that the project might have on other systems given its practical usage in research and development. That is, on the assumption that this project is used as a source to determine practical implementation of algorithms in *other* projects, then the effects are listed below.

Area	Description	Examples
Public health, safety, and welfare	Were it to be adopted in city planning or map traversal settings and effectively reduced travel time for numerous vehicles, it would contribute to reduced gas emissions, and therefore minimize the spread of CO_2 and other smog contributors which negatively impact public health.	• Reduce pollutant emissions on roadways. • Reduce per-vehicle time on the road alleviates traffic. • More efficient paths allow for a reduced first responder and emergency vehicle time-to-arrival.
Global, cultural, and social	A successful derivation of efficient network designs and roadways can allow for city planners to develop more refined layouts.	• Road layouts in areas such as London can be refined to reduce congestion. • A reduction in vehicle travel time improves driver and passenger quality of life.
Environmental	Implementing a more efficient path-finding algorithm can lead to significant improvements in code efficiency, as execution time and power consumption are closely linked. This, in turn, can reduce energy consumption, heat generation, etc.	• Reduction in power consumption for workloads tied to algorithm execution. • Reduced emissions of CO_2 and other pollutants from vehicles.
Economic	Users can improve their own products or designs by identifying a more efficient shortest path algorithm for their workload, reducing compute time. It can also cause network layouts and topologies to have cascading effects.	• Compute time is reduced for a given workload, therefore reducing server cost per unit of work. • City layout improvements results in citizens saving on the cost of fuel as well as having more time.

Table 4.1 - Broader Design Context

4.1.2 User Needs

The stakeholders for this project include educators, researchers, and casual users as defined in section 2.2. A rehash of their needs is provided below.

- Educators need a way to visualize shortest-path algorithms because it will help them teach students better since visualizations can be very useful in learning. They will also need a way to showcase the shortest-path algorithms' efficiency using empirical evidence because it will help them to teach how the efficiency of the algorithms compares with others.
- Researchers need a way to visualize shortest-path algorithms because it will aid them in understanding and visualizing their research.
- Casual users need a way to visualize shortest-path algorithms because it will help them calculate the shortest path for various uses like driving their car from one point to another.

4.1.3 Prior Work/Solutions

Details regarding the team's literature review and analysis of similar products are provided.

Relevant Background and Literature Review

At the start of the semester, the team focused on developing a taxonomy summary based on the contents of the paper titled "A Survey of Shortest-Path Algorithms" [4]. The document can be found on the team's Senior Design website.

Similar Products

Below is a list of existing products on the web that are similar in nature to our current project. These products all provide a visualization for many common single source shortest path algorithms, as well as a way to let users implement their own data sets to be tested on, albeit in a limited manner.

- *Pathfinding Visualizer* by Clément Mihailescu [5]
- *Single-Source Shortest Paths* by VisuAlgo [6]
- *Find Shortest Path* by Graph Online [7]

While these products provide a similar experience to ours, they do not overcome many of the deficiencies this project plans to address. Below is a list addressing the advantages and shortcomings of our project in relation to these previously existing products.

Advantages

- Can evaluate more complex data sets with unique properties (i.e., varying edge weights, large data sets)
- Provides empirical data for the user based on the algorithms' performances.
- Allows for the direct comparison of multiple algorithms.

Shortcomings

- Less meticulous visualization of algorithms at work.
- Not designed for users with no experience in single source shortest path algorithms
- Limited number of unique algorithms to work with

4.1.4 Technical Complexity

The project's complexity arises from the ambitious goal to develop a flexible visualization tool that enables users to integrate datasets and algorithms, evaluate their efficiency, and visualize them across multiple maps. The use case diagram, shown previously in Figure 2.1, illustrates this complexity, with further details below.

- The system should support multiple algorithms and datasets. This necessitates a scalable format for adding algorithms and datasets, as well as requiring data persistence.
- Algorithm execution must be measured, and metrics must be gathered regarding their runtime and space complexity for the purposes of comparison. The system must therefore use statistical analysis to gather and present this information to users in a usable manner.
- Parallel execution of algorithms to support multiple user workloads.

4.2 Design Exploration

The design process involved exploring the goals, constraints and tradeoffs inherent to the project; these can be found below.

4.2.1 Design Decisions

In accordance to IEEE 42010 section 6.10.1, key architecture decisions and rationale can be found below.

- Developing a web application is beneficial as it enables workload separation, multiple simultaneous requests and improving overall performance on lower-end devices by abstracting the main algorithm execution logic from the user-facing portion.
- Usage of a RESTful API came as a result of its flexibility. Namely, the robustness of the HTTP standard, the straightforwardness of defining API endpoints, and the ability to detach the frontend service from the backend entirely.
- A small set of shortest-path algorithms were selected to focus on the visualization and feasibility of the concept while providing a polished and refined user interface.
- Separating the algorithm execution logic from the REST API server component such that adding and removing the algorithms is not directly tied to the server itself, allowing for streamlined development.

4.2.2 Ideation

A detailed breakdown of our ideation process can be found in 9.2 Ideation Lotus Blossom Diagram.

4.2.3 Decision-Making and Trade-Off

Following the ideation process, the team proposed several UI interfacing tools. To determine which tool or tools to utilize in this project, a decision matrix was used to assist in the process, found in Table 4.2. Each tool was scored out of 5.

Criteria	Weight	CSS	HTML	JavaScript frameworks	Bootstrap	Foundation
Ease of use	3 *	3	4	3	4.5	3.5
Documentation	4 *	4	5	2	4	3
Learning Curve	4 *	4	5	2	4	3
Performance	5 *	5	5	3	4	4
Customization	5 *	5	4	5	4	4
Total	21	76	64	45	62.5	49.5

Table 4.2 - Decision-Making Matrix

Based on these conclusions, the team decided to use an integrated mix of user interface tools. Specifically, the team plans to format the UI with HTML and CSS, and then utilize Bootstrap for the web application to create an intuitive UI that is accessible on all devices. The decision was relatively streamlined, as HTML provides a dependable foundational structure for the UI, CSS is easy to use for extensive customization, and Bootstrap provides a modular approach with pre-built components that ease the development process.

4.3 Proposed Design

The team proposes a high-level design that takes into consideration the observations and decisions made in prior sections, alongside other meaningful details regarding the implementation, below.

4.3.1 Design Visual

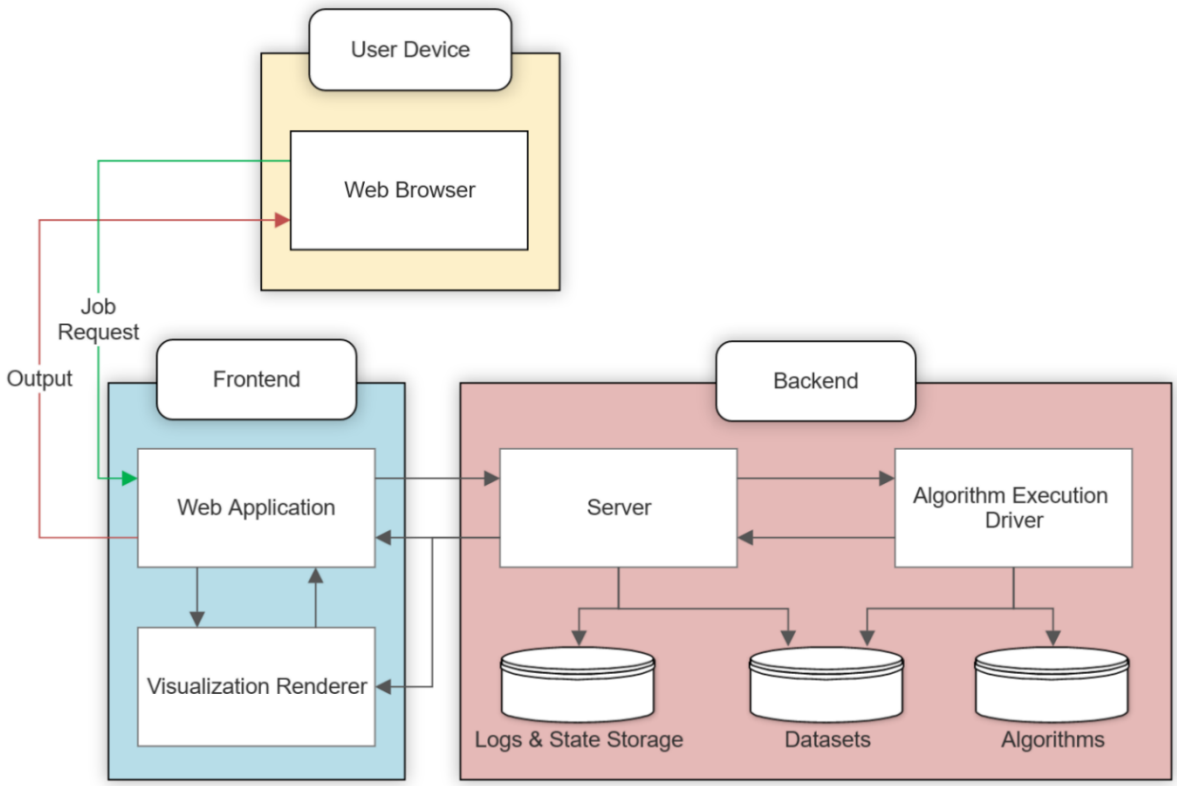
The design is split into two major parts that correspond to a traditional Model-View-Controller full stack application: a “Frontend” and “Backend”, with a visualization provided in Figure 4.1.

The “Frontend” has been divided into two logical sections.

- The *Web Application* is responsible for receiving and transmitting requests from the user to the server; this component is intended to be straightforward.
- The *Visualization Renderer* is a custom implementation for visualizing graph outcomes and rendering them as an image. This visual is then passed to the *Web Application* to display to the user.

Likewise, the “Backend” is also divided into two sections:

- The *Server* is responsible for managing RESTful requests from the *Web Application*, queueing requests for the *Algorithm Execution Driver*, and logging all relevant activities.
- The *Algorithm Execution Driver* is a separate component whose purpose is to translate the requests to commands applicable to each algorithm and dataset to manage their execution.



- Figure 4.1 - Block Diagram

The "User Device" section is only present for informational purposes; it is not a concern for the project.

4.3.2 Functionality

In the real-world, the product is intended to be used in the research stages of the user's projects. That is, if the user wants to implement an efficient shortest path algorithm, they can use the product to inform themselves on the candidate algorithm's benefits and downsides.

Scenarios

As captured in our use case diagram, here is one possible scenario:

> The user may choose two or more shortest path algorithms as the candidates for their road traversal software, and it may be the case that one is better than the other in certain cities or regions. Given that the user can submit a properly formatted dataset, the tool can provide a detailed comparison of these algorithms over those road networks, allowing the user to identify where to utilize each of them preemptively.

Additionally, the product is intended to facilitate academic research. One such scenario is presented:

> The user may develop variations of existing shortest-path algorithms in the search for optimal algorithms for specific uses, though it is possible that their variation results in a regression. Given that the user can include their implementation in a modified version of the tool and a relevant dataset, the tool allows for a detailed comparison between the original and the variation such that they can identify regressions before continuing their research.

Finally, the product is intended to serve as an aid for teachers and students alike:

> The teacher explains how a new algorithm works to a student. The student wishes to comprehend the differences between the original and the variation yet may not have the facilities to implement them. Given no additional information, the tool can present the user with pre-defined datasets and algorithms that they can choose to compare. The comparison includes a visualization of the path taken and metrics that can help the user understand what the differences are with minimal obstacles.

The product is designed to fulfill the needs in these scenarios while simultaneously addressing the requirements laid out in section 2.3.

4.3.3 Areas of Concern and Development

The team has identified some areas that need special attention, along with their potential impact, and what prevention plans have been idealized for them:

Potential Issues	Potential Impacts	Immediate Prevention Plans
The potential of providing incorrect information and the generation of the visualization is incorrect as a result.	Frustration and loss of confidence in the program by potential users.	Extensive testing of the algorithms provided and corroboration of their correctness with advisor.
The information provided by the algorithm is correct, but the visualization generated from it is not.	Confusion for the user arises from the discrepancy and potentially poor decision making in response.	Additional validation and testing of the visualization software provided.
The measurements of the efficiency of a particular algorithm are incorrect and it inaccurately reflects said algorithm.	The user picks the wrong algorithm for their current task, potentially wasting time and resources, which could in turn cause the failure of their project.	Generation of documentation and review of the provided algorithm code and their runtime equations.

Table 4.3- Potential Issues and Impact

5 Testing

The testing phase of a software project is critical to ensuring that it meets its stated requirements and quality standards. This section outlines the testing process for our software application, including the objectives, types of testing, testing methodology, resources, and expected results.

5.1 Testing Objectives and Tools

5.1.1 Objectives

The testing procedures are based on the proposed design of the system's components from section 4.3, and further rooted in the guidance provided in IEEE 29119 section 4.1.8. The main motivations for defining the testing procedures are:

- Identify any defects or issues that could impact the performance or usability of the system.
- Ensure that the requirements presented in section 2.3 are being met, and that deviations from it are recorded and documented.
- Validate the proper interaction between the various parts of the system, adhering to the best practices and industry standards for each component.

5.1.2 Tools

Tools for testing the project can be found below, alongside a brief description of what purpose they will fulfill.

Frontend	
Jest	Jest allow for testing functionality such as UI components and application logic.
Backend	
JUnit	Standard Java testing framework and dependency.
Spring Boot Test	Spring Boot provides support for testing applications with JUnit using the Spring Boot Test extension. In particular, it will serve to test the REST functionality of the server application.
Mockito	Mockito can be used in conjunction with other testing frameworks to create and inject mock objects that simulate database operations. This allows testing for the behavior of the application without interacting with the database.

Table 5.1 - Testing Tools

IEEE 29119 section 4.1.10 defines a test oracle as a source of information that determines whether a test passes or fails [2]. For this project, the oracles include the requirements and design specification laid out in section 2.3 and 4.3, respectively, and the project's advisor will serve as a guide for the algorithm execution metrics.

5.2 Unit Testing

Traditionally, unit testing focuses on the smallest testable components of a software system, such as functions, classes, modules, etc. For this iteration of the design document, the focus will be on unit testing the system components shown in Figure 4.1; further specificity will be included as the project reaches the implementation phase.

5.2.1 Frontend

The units defined for the frontend consist of the graphical user interface (UI), the Login Functionality, and the Visualization Renderer. The units will be implemented using JavaScript, and tested using the library Jest.

UI and Login Functionality

The first interaction with the UI will be for the user to enter the credentials and get access or register if it's the first time the system is used. The data entered (login and password) must match the values stored, which will be part of integration testing.

The UI will also provide options for the user to select input parameters and view the output of executing particular algorithms on particular datasets. The input specification will be enabled by dropdown menus allowing the user to select: (a) the input graph dataset and (b) the algorithm to execute. To test the input selection, a set of tests will ensure that whatever the user selects is properly stored so it can be passed to the backend; a similar test will done for the algorithm selection.

Finally, the UI will display the metrics attached to executing a particular algorithm, such as the runtime. Testing must also assert that values are correctly being passed to the frontend, such that they are accurate for any algorithm execution,

Visualization Renderer

To enable a richer set of features for the intended users, two external tools are being considered: Graphology and MapBox. Graphology will be used for more interactive datasets whereas MapBox will be used to display datasets which correspond to real cities.

Testing procedures for these external tools are as follows:

- Graphology: Ensuring that a small graph can be properly constructed based on an implicit specification of it and the visualization properly reflects the relationship between the edges and nodes.
- MapBox: Testing will focus on displaying the map of the city correctly, properly illustrating the streets and intersections, provided that the input datasets correspond to such settings.

5.2.2 Backend

There are two main components defined for the backend: the API server and the Algorithm Execution Driver (AED). Each of them will interact with three basic persistence methods seen in Figure 4.1 - Block Diagram.

The server component follows the *Controller* role in the Model-View-Controller design pattern, which serves primarily as an interface between the *Model* (AED) and the *View* (Web Application) roles. Therefore, the testing related to it is developed further in section 5.3.2.

Algorithm Execution Driver

The AED can be logically divided into units to be tested more thoroughly, focusing on several key aspects:

- The accuracy of the algorithm executions must be prioritized in testing; incorrect or misrepresented results undermine the main utility of this project.
    - The importance of this is reflected in the risks defined in section 3.5.
- The reliability of the system under abnormal situations must also be tested. The processing of various algorithms and datasets, some of which are user submittable, requires that a malicious input does not render the system unstable.
- Testing the validity of the datasets programmatically against their specified format using static analysis tools is required.

Since most of the AED's logic is implemented in Java, the JUnit library will be used for testing.

Algorithms

Algorithm tests are slightly more complicated as they are provided to the project by the advisor rather than being implemented by the team[3]. As such, there is a set of assumptions associated with both them:

- The provided algorithm implementations have been thoroughly tested independently from this project.
- Further testing for the sake of this project need only concern itself with the modifications made to the algorithm implementations rather than the implementation as a whole. This includes regression testing, detailed in section 5.6.

If these assumptions prove incorrect, each algorithm will be subject to a testing suite similar to that defined for the AED.

Logging & State Storage

Lastly, the backend component must provide a method for logging activity. Logs are a facility to determine the state of the system and its jobs, and therefore must be thoroughly tested. As previously mentioned, the server-side components are to be implemented in Java, so logging will utilize an industry-standard solution such as SLF4J or log4j. Testing for the storage and validity of these logs is therefore relatively straightforward; a check on the output of a method's logging will be performed on the target output file.

5.3 Interface Testing

Interface testing is an important part of the testing suite, as it helps ensure that the different parts of a system communicate with each other effectively.

[3] The objective of the project is to develop a proof-of-concept implementation. Therefore, the team and advisor will provide around three algorithms and a similar selection of datasets for the purposes of evaluating the system's functionality.

5.3.1 Frontend

For the frontend, there are a few different interfaces that will need to be tested throughout our project:

Login/Registration System

The login/registration system's connection to the backend server will need to be tested. The login system will need to connect to the backend server to find if the login information input by the user matches up correctly with a user account that exists. The registration system also needs to connect to the backend server so that the new login a user wants to use is saved and stored for future login attempts.

Algorithm Selection/Data Upload

The algorithm that the user selects from the dropdown menu and the data set they put in need to be properly stored so they can be sent to the backend. The selected algorithm needs to be sent to the algorithm execution driver so that it can perform the algorithm execution on the user's data set which is sent to the server.

Algorithm Execution Metrics

The algorithm execution metrics that are displayed to the user will need to connect to the backend server, which will retrieve the execution results from the algorithm execution driver. Testing will involve verifying the produced metric from the algorithm execution driver is correctly outputted to the frontend.

Visualization Renderer

The visualization renderer will need to retrieve the data set from the server-side storage to properly display them in the web application. Additionally, the visualization renderer must be connected to the algorithm execution driver, which computes the algorithms' results. Testing will involve ensuring the visualization renderer is able to successfully retrieve the data sets from the server and parse through them. It will also involve testing the connection to the algorithm execution driver and ensuring that the visualizations being produced coincide with what is being computed in the driver.

5.3.2 Backend

For the backend, the server has multiple responsibilities and interfaces with various system components. The three major connections are:

REST API Endpoints

The API server test suite will check correct HTTP responses to requests presented on each endpoint, such that their output aligns with what is expected in the web application component. This includes testing for edge cases and error scenarios to ensure that the server can handle unexpected input and return appropriate responses.

Persistence Layer

Persistence layer testing will ensure that data is properly stored, retrieved, and updated, and that the communication between the server and databases is functional.

Algorithm Execution Driver Controller

The AED will require specific commands to handle requests; the test suite will verify the correctness of these commands as issued by the endpoint requests. The format for algorithm execution results will also be tested before being returned to the web application component.

5.4 Integration Testing

Integration testing builds on the concept of interface testing by ensuring that the different components operate as expected together, thereby ensuring that the system works together.

In the context of the frontend, the key modules that must be integrated and tested are the login functionality, the algorithm execution metrics, and the visualization renderer. In the context of the backend, the key modules that must be integrated and tested are the REST API endpoints, the persistence layer, and the AED controller, all of which are part of the server component.

Login Functionality and Algorithm Execution Metrics

Tools: Jest

- Test that the metrics for a user's previous algorithm executions are stored in the account attached to their login.
- Verify that multiple different metric reports can be stored in the user's account.

Algorithm Selection/Data Upload and Algorithm Execution Metrics

Tools: Jest

- Test that the algorithm the user selects and their uploaded data set displays the correct execution metrics, such as runtime.

Algorithm Execution Metrics and Visualization Renderer

Tools: Jest

- Test that the metrics for each algorithm execution are displayed alongside visualizations after the user runs an algorithm on their data set.
- Verify that the metrics and the visualization are stored in a report for the user.

Login Functionality and Visualization Renderer

Tools: Jest

- Test that the visualizations created for a user's algorithm execution are stored in their account attached to their login.
- Verify that multiple different visualizations can be stored in the user's account.

API Endpoints and Persistence Layer Integration

Tools: Mockito, Spring Boot Test

- Test the creation, retrieval, update, and deletion of datasets through API endpoints.
- Verify that proper error handling is in place when attempting to perform invalid operations (e.g., updating a non-existent record).
- Test data consistency and persistence through API calls.
- Test that only user-relevant data is being retrieved and transmitted.

API Endpoints and AED Controller Integration

Tools: Spring Boot Test

- Test the successful execution of algorithms, verifying that the correct inputs are being passed from the API endpoints to the AED Controller.
- Verify that the correct execution results are returned to the API server in the expected format.

AED Controller and Persistence Layer Integration

Tools: Mockito, Spring Boot Test

- Verify that the AED controller can retrieve and store algorithm execution results and logs on the persistence layer.
- Verify that the stored results reflect the measured algorithm results.

End-to-end Backend Integration

Tools: JUnit

This is a combination of the following integration tests:

- API Endpoints and Persistence Layer Integration
- API Endpoints and AED Controller Integration
- AED Controller and Persistence Layer Integration

The purpose is to ensure that all the different components of the backend are properly integrated and working together as expected. These tests will verify that operations originating from the API endpoints are completed in their entirety, including persistent storage operations and algorithm execution requests.

5.5 System Testing

System-level testing is the highest level of testing, which relies on the lower-level testing strategies previously detailed in this document. The primary objective of this testing is to validate the system's ability to meet the requirements specified for this project and to demonstrate the thoroughness of the test suite.

Table 5.2 shows a mapping of which set of tests address certain functional requirements of the project, derived from the identified use cases featured in prior sections. Each row can be treated as a system-level test, wherein each column lists the testing target for the given type, and where testing targets correspond to the ones detailed in previous sections.

Requirements	Unit Tests on...	Interface Tests on...	Integration Tests on...
Availability of multiple algorithms and datasets	Algorithm Execution Driver	Algorithm Execution Driver Controller	AED Controller and Persistence Layer Integration
Provide correct measurements for runtime	Algorithm Execution Driver; Algorithms	Algorithm Execution Driver Controller	Algorithm Selection/Data Upload and Algorithm Execution Metrics Integration
Distinguish algorithms based on edge weights	Algorithms	--	--
User can select datasets and algorithms to test	UI and Login Functionality; Algorithms	Algorithm Selection/Data Upload	API Endpoints and Persistence Layer Integration
Generate informative visuals; allow for comparison & export	UI and Login Functionality; Visualization Renderer	Algorithm Execution Metrics; Visualization Renderer	Algorithm Execution Metrics and Visualization Renderer Integration

Table 5.2 - Functional Requirements System Tests

5.6 Regression Testing

Following the completion of system testing, further additions to datasets will be supplemented with tests to ensure no loss of responsiveness and functionality is inflicted on the UI and algorithm visualization. This includes ensuring that previous datasets are tested and provide a near-equivalent result to before the changes. Another part of this is ensuring that prior algorithms will work with new datasets (and vice versa, any new algorithms will work with the existing datasets) while providing accurate runtime feedback.

5.7 Acceptance Testing

As part of the agile development process, the prioritizes acceptance testing each sprint to meet the client's needs while adhering to specific criteria:

- UI Response not exceeding 5 seconds after a particular algorithm for determining a shortest path has completed its execution.
- Each component of the UI is easy to use with clearly defined purposes.
- Algorithm executions are on par with expectations.

Additional testing may be included based on the client's feedback at the end of each sprint, until the final design meets the accepted criteria.

5.8 Security Testing

Given our project is a proof of concept, security is not a major concern. In the aim of keeping up with industry practices, a stretch goal will include utilizing SSL for securing communication between the server and users. This will not be a major consideration in the event that other developments take priority.

5.9 Results

Testing results are not yet available as testing has not commenced.

6 Implementation

Alongside finalizing all the detailed design decisions and testing plans throughout this semester, the team was proactive and attempted to implement certain portions as well as prepare better for future implementation of the rest of the system. To that end, the team has reported on two main activities in the next two sub sections.

6.1 Implementation of the Shortest Path Algorithms

The team familiarized itself with shortest-path algorithm code that could be integrated in the final system as listed in the bibliography. The listed shortest-path algorithm code [12] has implementations of multiple shortest path algorithms like Dijkstra's, A*, CH, etc.

6.2 UI & Wireframe

In order to get a better grasp of the future UI implementation and save time in the second semester the team decided to generate a wireframe prototype seen below in Figure 6.1 as an overview of the UI. The tool used for this purpose was Figma, and an interactive version of the prototype can be found at the following link: User Interface Prototype (figma.com).

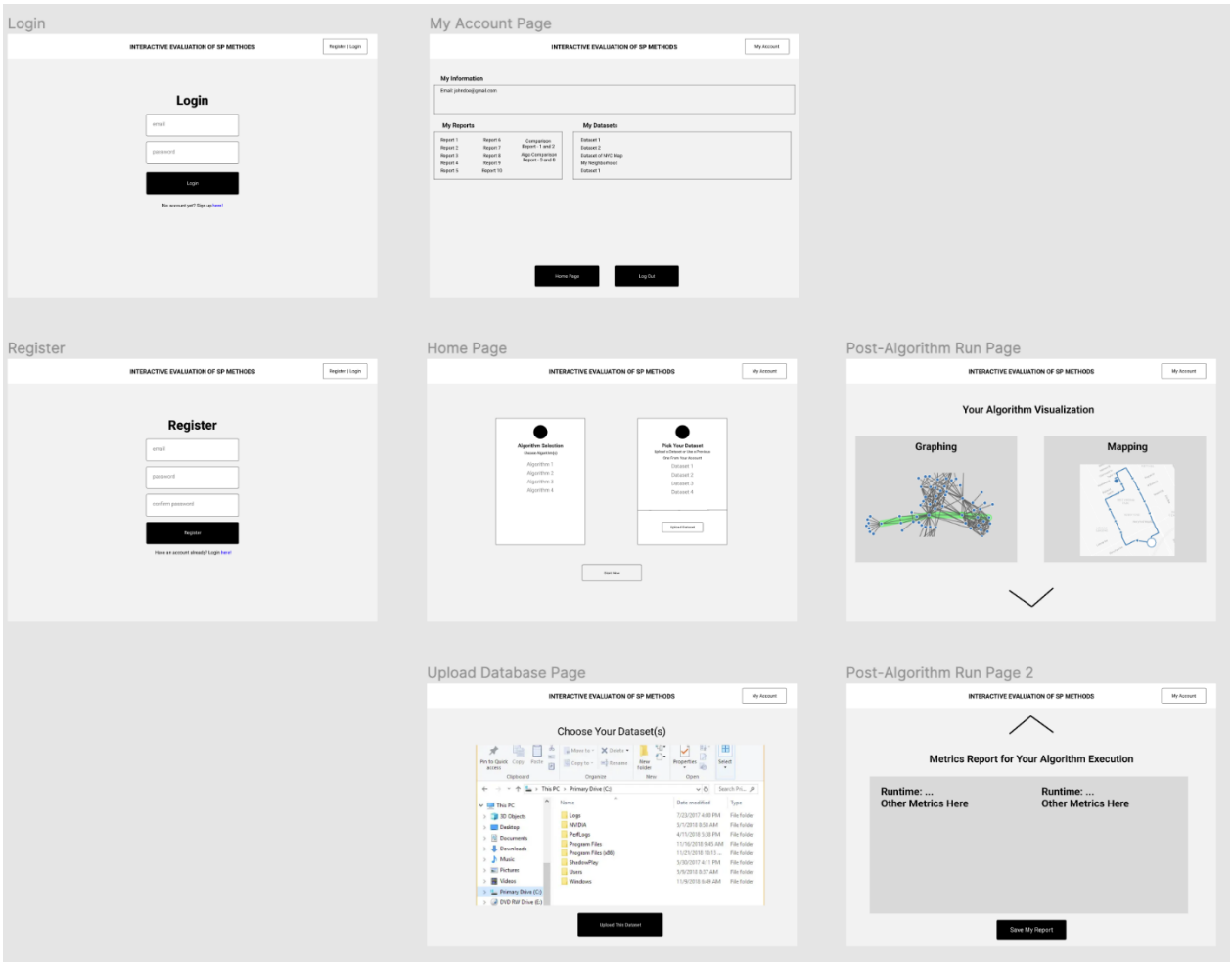


Figure 6.1 - Wireframe Prototype

In addition, the team familiarized itself with currently available solutions for algorithm visualization that could be integrated into the final system; namely, Graphology for node visualization [8], Awesome-Vector-Tiles [9], two of the several MapBox variations written in Python [10] and JavaScript [11].

7 Professionalism

This section describes the team's commitment to professionalism throughout the lifespan of the project. Specifically, it contains our understanding of how our actions and this project should align with NSPE's and ACM's software engineering code of ethics.

7.1 Areas of Responsibility

The areas of responsibility listed below are in relation to "Computer society and ACM approve software engineering code of ethics" [13].

Area of Responsibility	ACM S E Code of Ethics	In Our Own Words...
Work Competence	**Product:** Software engineers shall ensure that their products and related modifications meet the highest professional standards possible. **Profession:** Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.	The S E code of ethics is clear regarding the engineers' qualifications. There, they state that an engineer should have their mind on delivering quality products in a way that improves the perception of "software engineering" and support the public's interest. While NSPE specifies that one should "avoid deceptive acts", the S E code of ethics suggests that we should work in favor of advancing the profession.
Financial Responsibility	**Client and employer:** Software engineers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest. **Product:** Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.	The S E code of ethics suggests that the engineer act in accordance with what benefits the employer and the public. It matches the NSPE definition reasonably well, but it makes no mention of financial responsibility. That said, supporting the employer can also include this capacity.
Communication Honesty	**Client and employer:** Software engineers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest. **Colleagues:** Software engineers shall be fair to and supportive of their colleagues.	The S E code of ethics is a bit different compared to the professionalism paper when it comes to working with colleagues. The S E code of ethics states that you must be fair and supportive to your team members while the professionalism paper has more of an emphasis on telling the truth and not being deceptive.
Health, Safety, Well-Being	**Self:** Software engineers shall participate in lifelong learning	The S E code of ethics states that engineers work to promote ethical

	regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.	approaches to practice in the profession and extends this to the reputation and interests of the public. The NSPE definition holds similarly by suggesting holding the health safety and welfare of the public above all
Property Ownership	**Management:** Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance. **Colleagues:** Software engineers shall be fair to and supportive of their colleagues.	Engineers should honor all forms of intellectual property owned by the employer. They do not have the right to profit from independent sales or use of their intellectual property.
Sustainability	**Self:** Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession **Client and employer:** Software engineers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest.	The S E code of ethics does not have anything specific about the environment unlike the professionalism paper. Although the environment specifically is not mentioned, the code of ethics states that engineers must operate ethically and, in the public's, best interest.
Social Responsibility	**Public:** Software engineers shall act consistently with the public interest. **Profession:** Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.	Engineers must prioritize society's benefit, while also upholding their company's and profession's reputation. The NSPE canon is more detailed than the ACM Code of Ethics in this regard, specifying that engineers should behave honorably, responsibly, ethically, and lawfully to improve the profession's reputation and usefulness.

Table 7.1 - Areas of Responsibility

7.2 Project-Specific Professional Responsibility Areas

The following is an outline of the specific professional responsibility areas that are relevant to our project.

- **Work Competence:** This has a high priority in our project because we strive to deliver a high-quality product that is successful in visually displaying algorithms and picking the best algorithm for our user's data sets. We want to meet the highest professional standards possible so that our client and users are satisfied.
- **Financial Responsibility:** This has a low priority for us because this is a relatively low-cost project, so finances and budget are not much of a concern.
- **Communication Honesty:** This has a high priority for us because it enables us to collaborate and communicate better with our client and amongst ourselves. Good communication and collaboration between everyone within a team is essential to the team's success.
- **Health, Safety, Well-Being:** This has a medium priority because we currently have a small group of target users, but in a future where we expand, this can help the public at large. It is not high priority now at this moment, however.
- **Property Ownership:** This has a medium priority because implementations of algorithm visualizers already exist and are publicly available, but the way we implement our app algorithm visualizer will be ours. Our backend will use pre-existing algorithms, and our frontend will use pre-existing visualizer tools, but the way we implement it will ultimately be unique.
- **Sustainability:** This has a low priority because our app will not have a big impact on the environment and is not something that would be ethically questionable.
- **Social Responsibility:** This has a high priority because our users (and potentially the public) should benefit from our tool. Algorithm visualizers are important for things like electric vehicles that require algorithms for shortest path calculations and need a way to visualize the shortest path for the user.

Most Applicable
Work Competence
This high-priority professional responsibility covers multiple categories, from testing and software integration to team planning and communication. Thus, the effects of this one can prove to be costly and could strain the overall project scope if not met. This creates a need to take precautions and plan the development of high-quality code, be it by preparing an extensive software tests suite to project planning and meeting coordination. To address these issues, our team has opted for two weekly meetings outside of our advisor meeting and peer reviewing our design documentation and code before submission. This will allow us to keep a better eye on the quality of the project and meet both the requirements and expectations.

8 Closing Material

8.1 Discussion

Given that the second term of Senior Design has not begun, the team has no results to report on.

8.2 Conclusion

Given that the second term of Senior Design has not begun, the team has no results to report on.

8.3 Team Contract

Team Meeting Schedule

Day	Time	Location	Preferred Methods of Communication
Tuesday	4pm – 5pm	Face-to-face, Parks Library	For updates, issues, scheduling, etc.: • Discord (for informal team communication) • Zoom (for client meetings) • Email (for formal communication) Otherwise, face-to-face meetings where possible for additional collaboration and discussion.
Thursday	2pm – 3pm	Client meeting, Zoom	
Thursday	3pm – 4pm	Team meeting, Discord	

Table 8.1 - Team Schedule

Decision-making Policy

Decisions will be decided via group consensus. For time-sensitive decisions, a majority vote will take precedence. Members absent from the decision-making process shall be notified immediately.

Procedures for Record-keeping

Minutes will be written after each meeting in an online document, recorded by Selma Saric. All documents produced for this project will be collected into a portfolio at the end of the semester.

Participation Expectations

The goal for weekly meetings is to gain a deeper understanding of the project, our assignments, and the client's expectations. As a result, they are extremely important and must be taken seriously. It is for that reason that each team member will be expected to attend all scheduled meetings unless an extenuating circumstance occurs. For conflicts, the team member is responsible for giving a 2-hour notice (where possible) so that the team can work around it.

Furthermore, each team member is expected to remain in active contact with teammates between the weekly group meetings and distribution of work. Tasks that have a defined deadline must have progress shared with the team the day prior to the following client meeting. In the face of difficulties, they are expected to keep in contact and seek assistance to ensure the punctual submission of work.

In short, each team member is expected to follow through on decisions the team as a whole make, complete their agreed-upon tasks, and communicate regularly for the benefit of the project.

Team Goals for Senior Design I

- Complete all relevant technical documents such that they serve as a complete, quality project definition.
- Use the aforementioned documents to strategize Senior Design II's semester outlook with expected task completion deadlines and client approval.

Assistance & Recognition

Team members will make themselves available to help the other team members if they are stuck or struggling with a certain task, to the extent that they are able. The team should seek to help each other when they have the means to do so and when given sufficient time.

Each member is incentivized to highlight their accomplishments and contributions in each weekly report. Their work will be attributed to them when communicating with staff, faculty, and third parties if the discussion warrants it. Likewise, work done in tandem with other team members should be credited to all participating members. The project should be a highlight of each member's contribution and how they resulted in the grand outcome.

Designated Roles

Name	Roles
Alex Blomquist	Setup & coordination of design documentation
Samuel Caldwell	Component design & testing
Selma Saric	Project management, minutes recorder, meeting coordinator
Yadiel Johnson	Document organization, review, & submission,
All team members are expected to contribute to client interactions, design documentation, and other team-centric tasks.	

Table 8.2 - Team Member Roles

Skills & Experiences

Name	Roles
Alex Blomquist	Software development experience, operating systems, shell scripting, software project management, embedded systems, and full stack development utilizing Java (Spring Boot and Android Studio)
Samuel Caldwell	Software development experience (Java, C/++, Python), software project management, software architecture design, frontend development (JavaScript, HTML, Kotlin)
Selma Saric	Software development experience, project/product management experience, web development experience, software architecture design experience, frontend development experience
Yadiel Johnson	Software development experience (Java, C++), server deployment & provisioning, databases (relational & graph-based), Docker CLI, backend w/ Spring Boot, Linux/Unix management, network communications & APIs

Table 8.3 - Team Member Skills

Inclusion

The team is responsible for ensuring a respectful and safe working environment for all team members to share their ideas and insights freely. When deemed necessary, each team member should provide constructive criticism for the team and suggest solutions for the problems they identify. That said, there must be clear communication between *all* team members. By seeking each member's participation in discussions, we will keep a continuous flow of ideas while also keeping every member in the conversation. We will avoid, to the extent possible, a singular member overtaking or single-handedly managing the conversation points or decision-making process.

Work Distribution & Management (Individual and Teamwork)

The team will try to assign each assignment or task according to what strengths and preferences are listed in the Team Initiation document. These tasks will be recorded and tracked with a project-centric calendar service (e.g. Trello) to stay on top of our assignments and their due dates.

Violations to the Team Contract

1. How will you handle infractions of any of the obligations of this team contract?
    - The severity of each infraction will be weighed independently, but each member reserves the right to directly communicate with the offending member.
2. What will your team do if the infractions continue?
    - The team will look at the situation that prompts the infraction and determine a meeting time where it will be discussed.
    - Continued negligence regarding the project may see escalation to the active Senior Design I instructor or other relevant parties.

I hereby state that:

a) I participated in formulating the standards, roles, and procedures as stated in this contract.
b) I understand that I am obligated to abide by these terms and conditions.
c) I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.

Signatures:

1)	Alex Blomquist	Date:	2/16/2023
2)	Selma Saric	Date:	2/16/2023
3)	Sam Caldwell	Date:	2/16/2023
4)	Yadiel Johnson	Date:	2/16/2023

9 Appendices

Material that was too large to be included in the main content or otherwise serves as an addendum can be found here.

9.1 Project Schedule Gantt Charts

Display Week: 1

TASK	PROGRESS	START	END
Phase 1: Research and Planning			
Discover Phase | Research	100%	2/14/23	2/14/23
TeamThink Constellation	100%	2/14/23	2/14/23
Phase 2: Documentation			
Team Initiation Assignment	100%	2/14/23	2/19/23
Professionalism Assignment	100%	2/20/23	2/26/23
Requirements, Constraints, and Engineering Standards	100%	2/27/23	3/5/23
SD Team Website V1	100%	3/6/23	3/12/23
Project Plan Assignment	50%	3/13/23	3/26/23
Design Assignment	0%	3/27/23	4/2/23
Testing Assignment	0%	4/3/23	4/9/23
SD Team Website V2	0%	4/10/23	4/23/23
Phase 3: Finishing Up			
Final Design Document	0%	4/10/23	4/23/23
Faculty Panel Presentation	0%	5/3/23	5/3/23

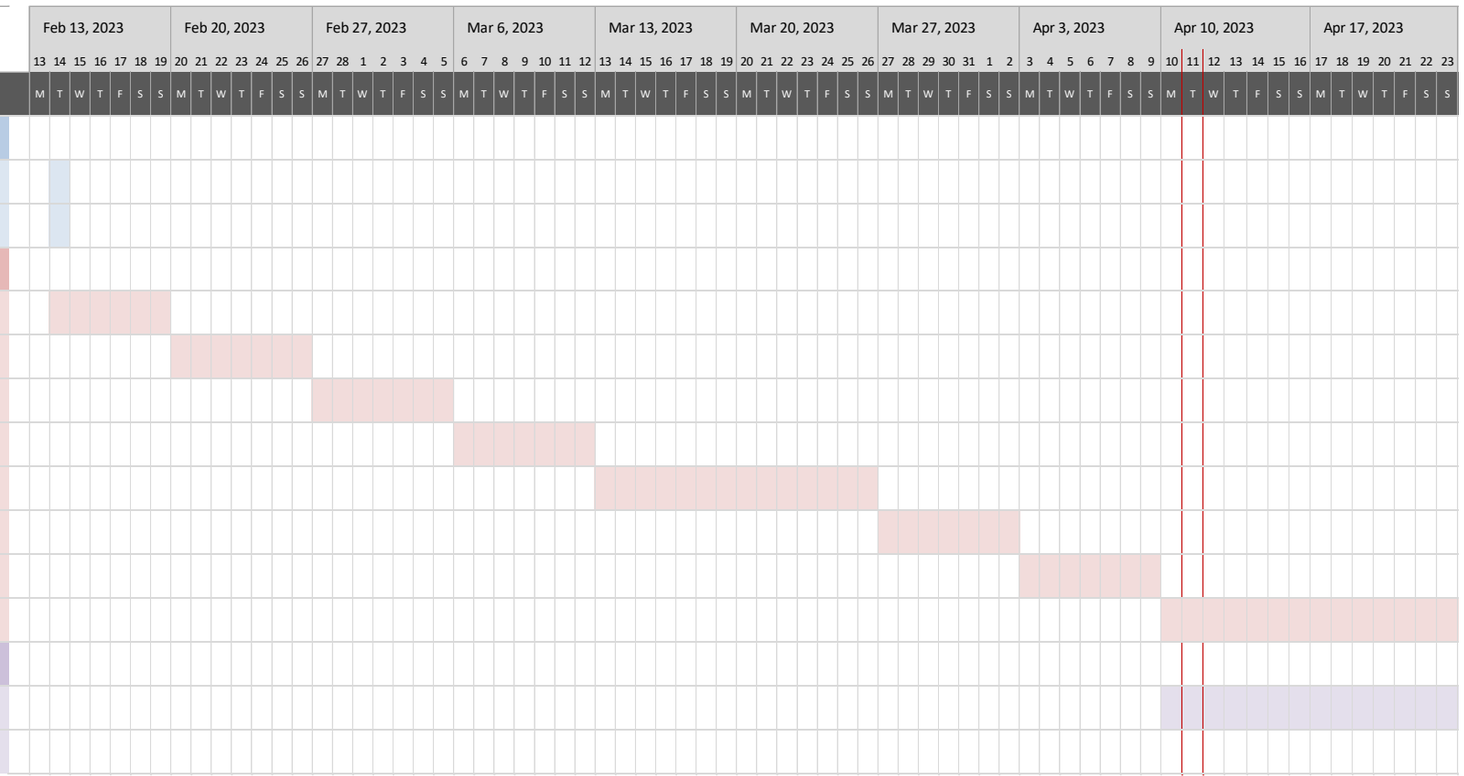


Figure 9.1 - Schedule, Semester #1

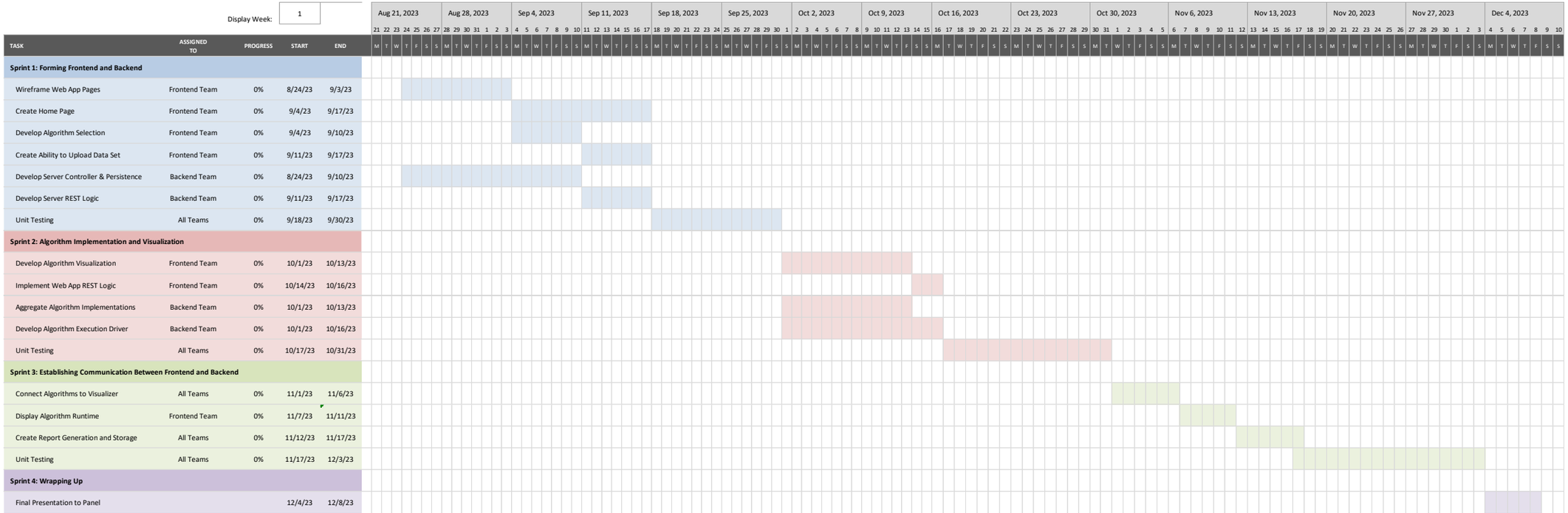


Figure 9.2 - Schedule, Semester #2

9.2 Ideation Lotus Blossom Diagram

Figure 9.3 - Lotus Blossom Diagram

10 Bibliography

[1] IEEE, "ISO/IEC/IEEE International Standard - Systems and software engineering -- Design and development of information for users," *ISO/IEC/IEEE 26514:2022(E),* pp. 1-76, 2022.

[2] IEEE, "ISO/IEC/IEEE International Standard - Software and systems engineering --Software testing --Part 1:General concepts," *ISO/IEC/IEEE 29119-1:2022(E),* 2022.

[3] IEEE, "IEEE/ISO/IEC International Standard for Software, systems and enterprise--Architecture description," *ISO/IEC/IEEE 42010:2022(E),* 2022.

[4] A. Madkour, W. G. Aref, F. U. Rehman, M. Abdur Rahman and S. Basalamah, "A Survey of Shortest-Path Algorithms," *arXiv preprint,* 2017.

[5] C. Mihailescu, "Pathfinding Visualizer," December 2016. [Online]. Available: https://clementmihailescu.github.io/Pathfinding-Visualizer/. [Accessed 2023].

[6] S. Halim, "Single-Source Shortest Paths," [Online]. [Accessed 2023].

[7] Graph Online; UnickSoft, "Find Shortest Path," 7 January 2023. [Online]. [Accessed 2023].

[8] G. Plique, *Graphology, a robust and multipurpose Graph object for JavaScript,* : Zenodo, 2023.

[9] T. MacWright, A. and e. a. , "awesome-vector-tiles," MapBox, 30 Aug. 2015. [Online]. Available: https://github.com/mapbox/awesome-vector-tiles.

[10] M. Perry, S. Gillies and e. a. , "mapbox-cli-py," MapBox, 11 May 2017. [Online]. Available: https://github.com/mapbox/mapbox-cli-py.

[11] J. Firebaugh, V. Agafonkin and e. a. , "mapbox-gl-js," MapBox, 23 June 2013. [Online]. Available: https://github.com/mapbox/mapbox-gl-js.

[12] yzengal, "sspexp_codes," BDA Group, 30 Apr. 2020. [Online]. Available: https://github.com/BUAA-BDA/sspexp_clone/tree/master/sspexp_codes.

[13] D. . Gotterbarn, K. W. Miller and S. . Rogerson, "Computer society and ACM approve software engineering code of ethics," *IEEE Computer,* vol. 32, no. 10, pp. 84-88, 1999.

[14] J. McCormack, S. Beyerlein, D. Davis, M. Trevisan, J. Lebeau, H. Davis, S. Howe, P. Brackin, P. Thompson, R. Gerlick, M. J. Khan and P. Leiffer, "Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment," 2012. [Online]. [Accessed 2023].

[15] G. F. I. Camil Demetrescu, "Dynamic shortest paths and transitive closure: Algorithmic techniques and data structures,," *Journal of Discrete Algorithms,* vol. 4, no. 3, pp. 353-383, 2006.

[16] H. Bast, "Car or Public Transport—Two Worlds," in *Efficient Algorithms: Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*, vol. 5760, S. Albers, H. Alt and S. Nher, Eds., Berlin, Springer Berlin Heidelber, 2009, pp. 355--367.