SDDEC23-14

ALEX BLOMQUIST
SELMA SARIC
SAMUEL CALDWELL
YADIEL JOHNSON

Interactive Evaluation of Shortest Path Methods

Problem Statement

Algorithm research is always developing, and efficiency is important, but hard to compare

This project aims to develop a system that enables:

- The use of Shortest Path algorithms namely All Pairs and Single Source.
- The use of different datasets like Escalon's road network pictured in Figure 1

...to output detailed comparisons in road-like networks for an educational settings

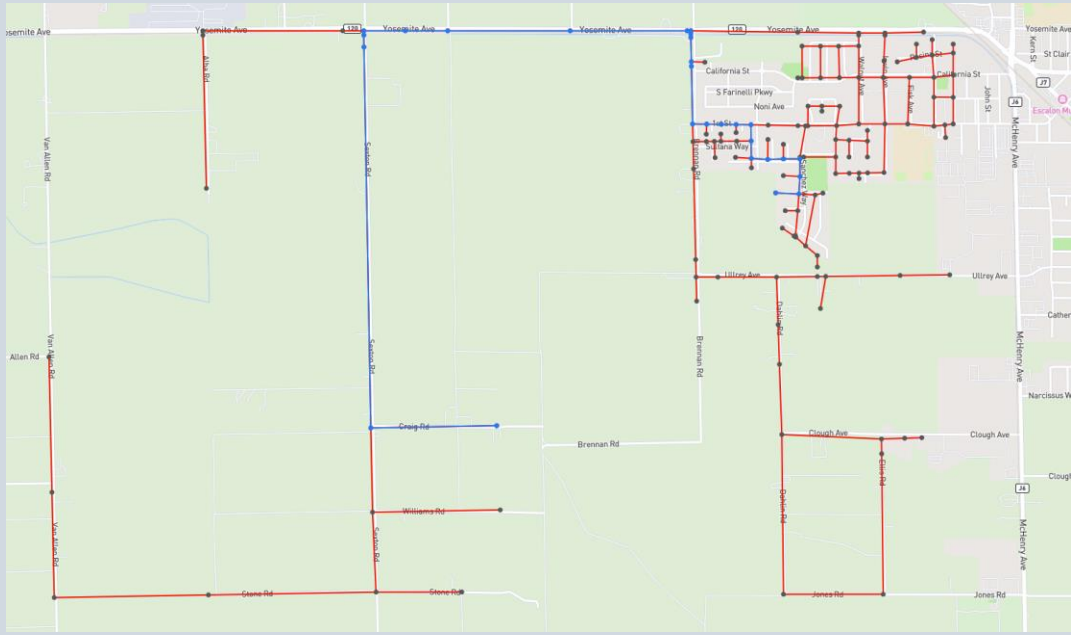


Figure 1: Shortest Path Visualized Route in Escalon, California Via MapBox

Stakeholders & Use-Cases

Educators:

- Present and educate people about the efficiency of different shortest-path algorithms

Students:

- Tool to better understand and learn about the performance of algorithms on different data sets

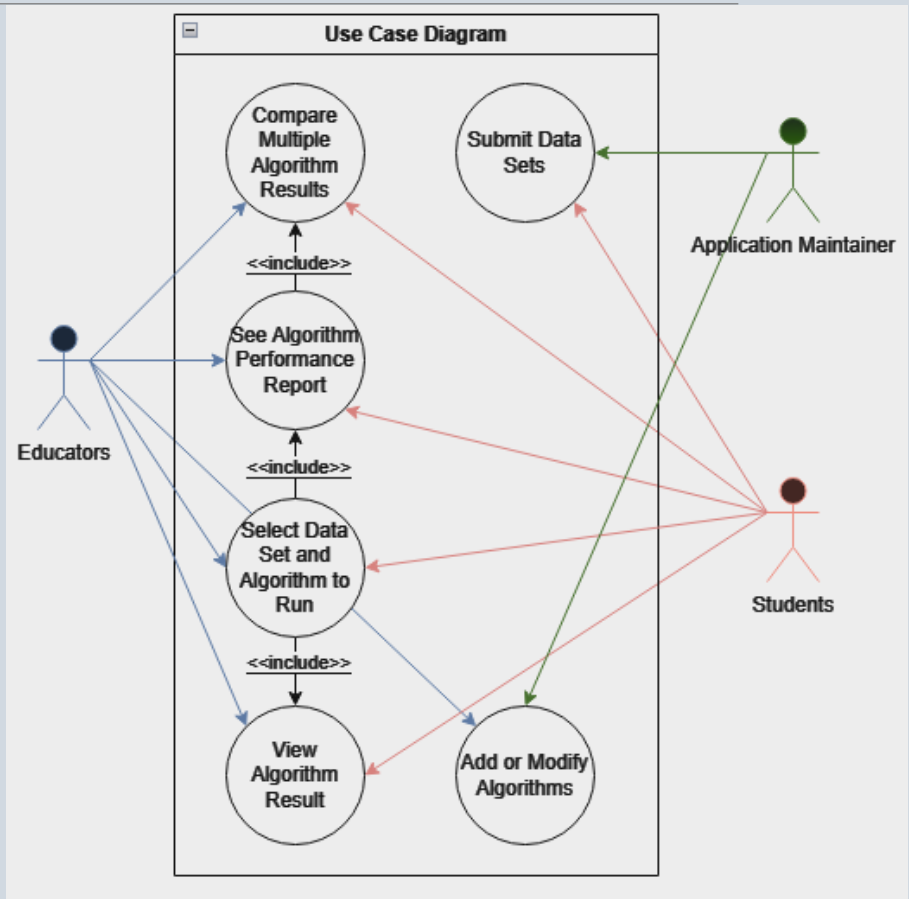


Figure 2: Use-Case Diagram

Requirements & Constraints

Requirements:

- User upload/algorithm selection
- Clean, organized presentation of SP visualizations
- Algorithm execution on data sets + metrics report
- Visualizations of algorithm outcomes/comparisons
- Optimal resource usage per algorithm run
- Report generation and storage

Constraints:

- Full-Stack Solution
- Budget: No more than $200

Initial Milestones

Milestone	Metrics:
Finalize System Architecture Design	(April 2nd)
Develop Server, Driver, and Web App Components	(Sept. 17th/Oct. 1st)
Algorithm Visualization/AED	(Nov. 1st)
Fully Develop User Interface/Server	(Nov. 11th)
Integration and System Testing	(Nov. 17th)
Final Software Release and Presentation to Panel	(Dec 3rd/Dec. 8th)

Original Design

Frontend

- Not many components on page
- Features were not organized

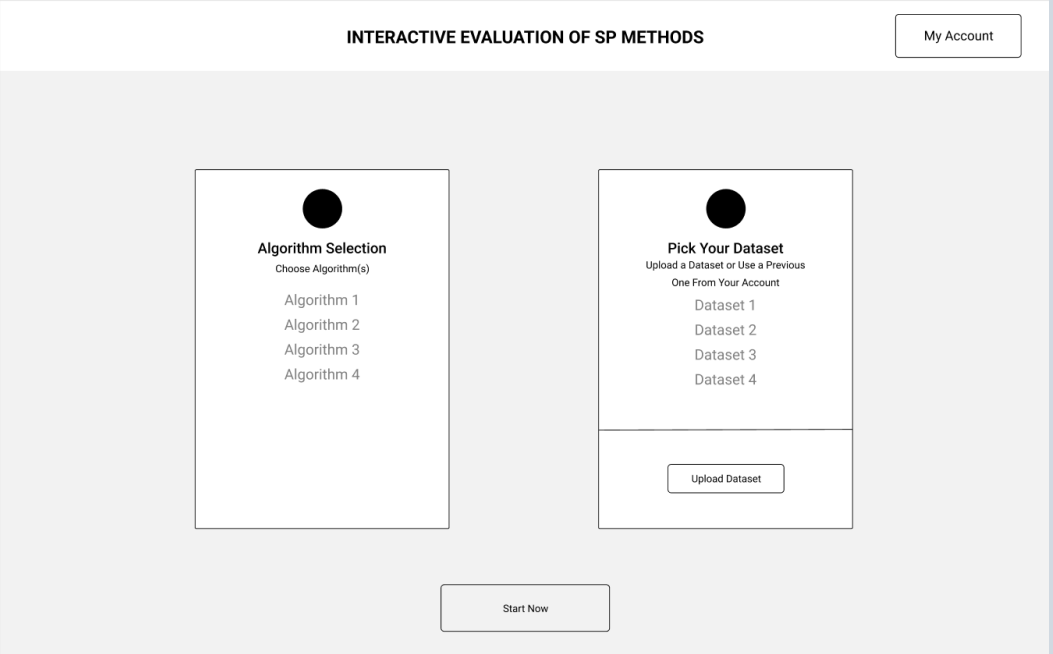


Figure 3: Original Home Page Wireframe

Original Design

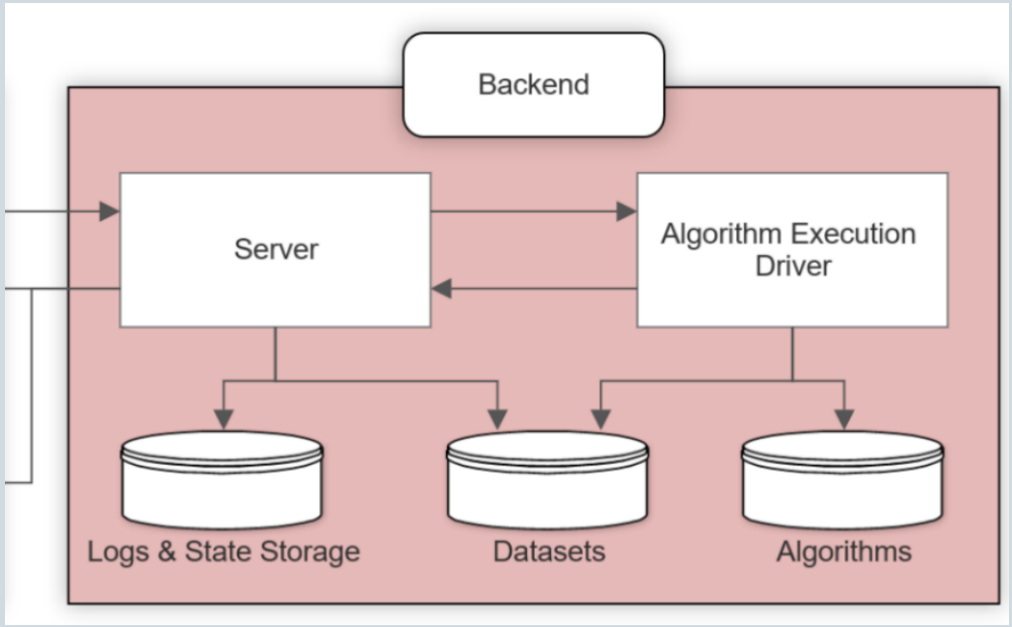


Figure 4: Original Backend Block Diagram

Web Server

- Serves web application
- Tracks algorithm executions
- Manages dataset storage

Algorithm Execution Driver

- Manages execution logic
- Multi-language
- Executes algorithms in C
  - Control logic in Java

How Our Design Evolved

Frontend

- Addition of tutorial page
- Addition of source/destination point selection
- User account removal

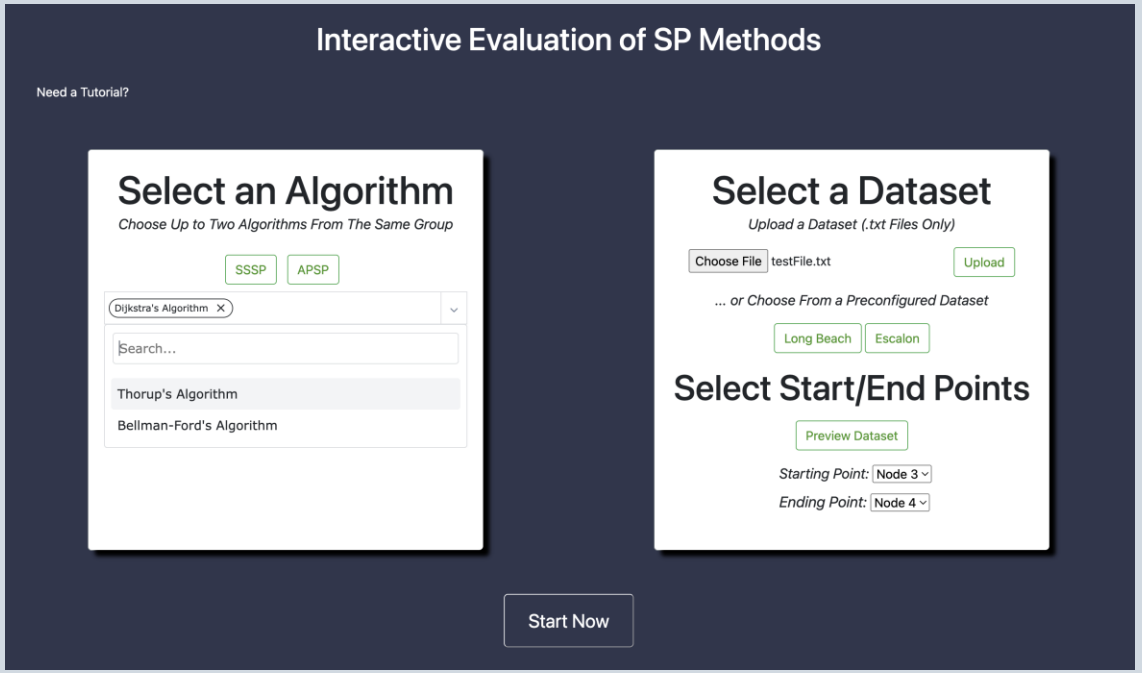


Figure 5: Updated Home Page

Issues with the Original Design

Issues

- Limited visibility
- Violations of requirements
- Split development
- Opportunity cost to remedy concerns

Revision

- Single language for backend development

Consequences

- Correlating space complexity to memory usage becomes implausible
- Alternatives require revising the entire architecture of the system

Evolved Design

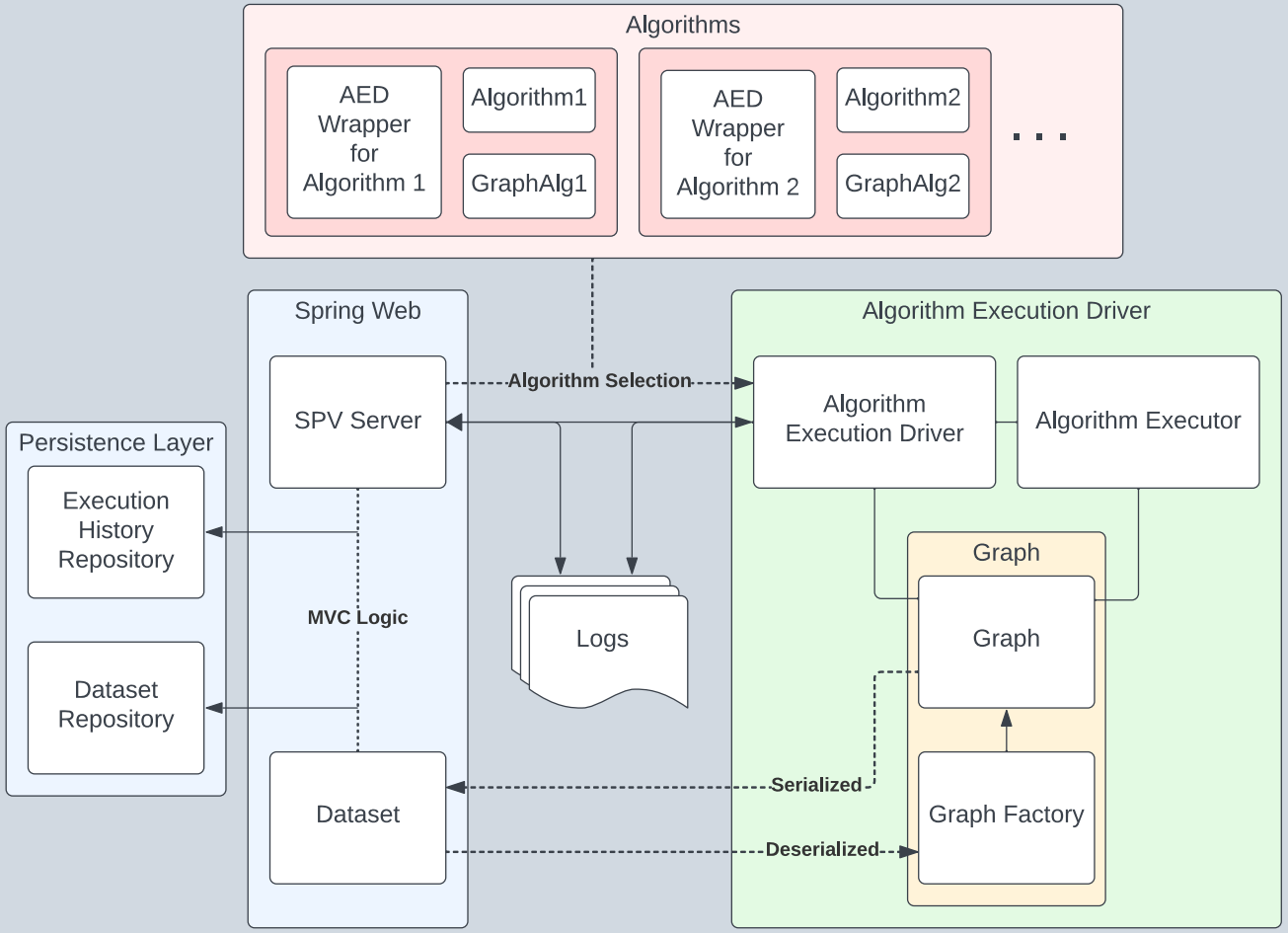


Figure 6: Updated Block Diagram

Implementation – Backend

Benefits of the Evolved Design

- Robust API
- Modularity
- Structured testing
- Language/Framework features

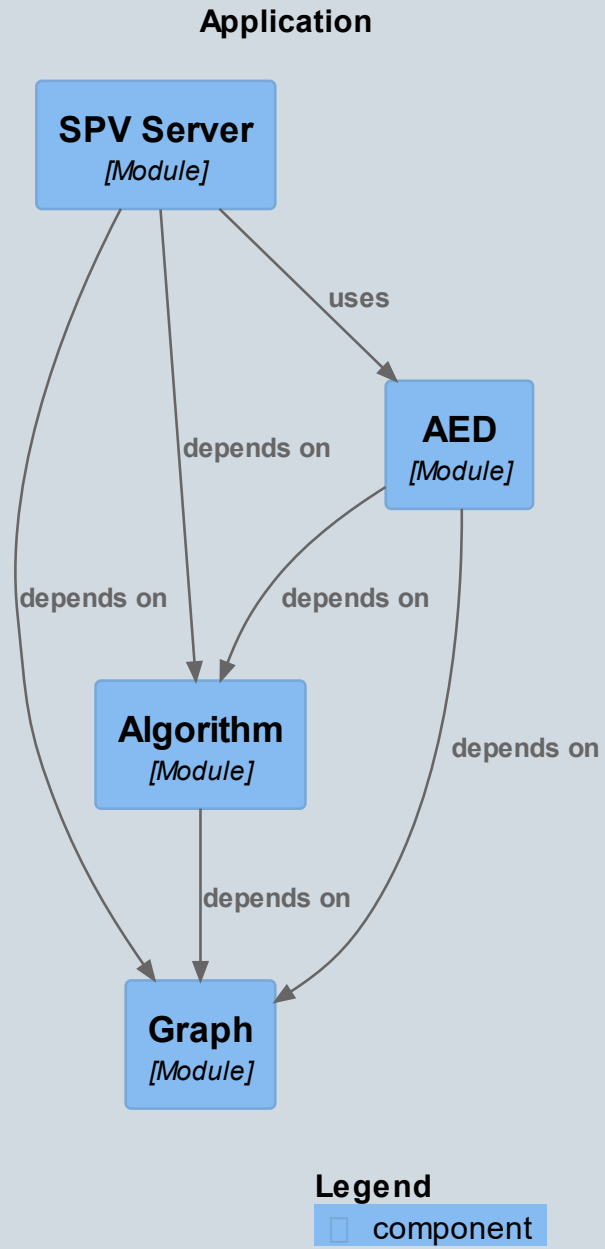


Figure 7: Module Dependency Diagram

Implementation – Backend

Benefits of the Evolved Design

- Robust API
- Modularity
- Structured testing
- Language/Framework features

POST

…/api/algorithm/apsp/evaluate/1?datasetID=1


```
{
    "information": {
        "type": "apsp",
        "predecessors": [···
        ]
    },
    "individualMetrics": [···
    ],
    "name": "Floyd-Warshall's Algorithm",
    "averageRuntime": 149.62,
    "runtimeRange": [
        145,
        163
    ],
    "totalRuntime": 7481,
    "standardDeviation": 3.554658914720229,
    "Percentile50th": 149,
    "Percentile75th": 150,
    "Percentile25th": 147
}
```


Figure 8: API Example

Implementation – Backend

Key Frameworks and Technologies Used

Web Server
- Spring MVC + Web

Persistence
- Spring Data JPA
- MySQL

Testing
- Spring MockMVC
- JUnit
- Jackson

Code Structure
- Spring Modulith

Development Operations
- GitLab Actions
- Docker

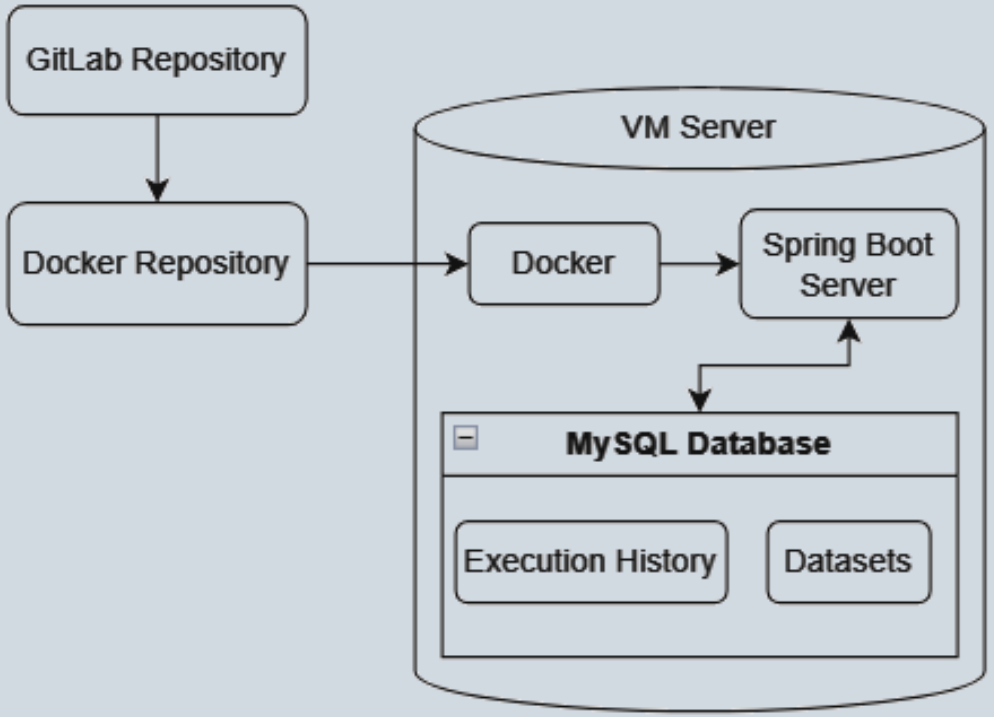


Figure 9: Deployment Strategy

Implementation - Frontend

- Homepage
  - Mapbox Visualization Page
  - Sigma Visualization Page
  - Results Page
  - Tutorial Page

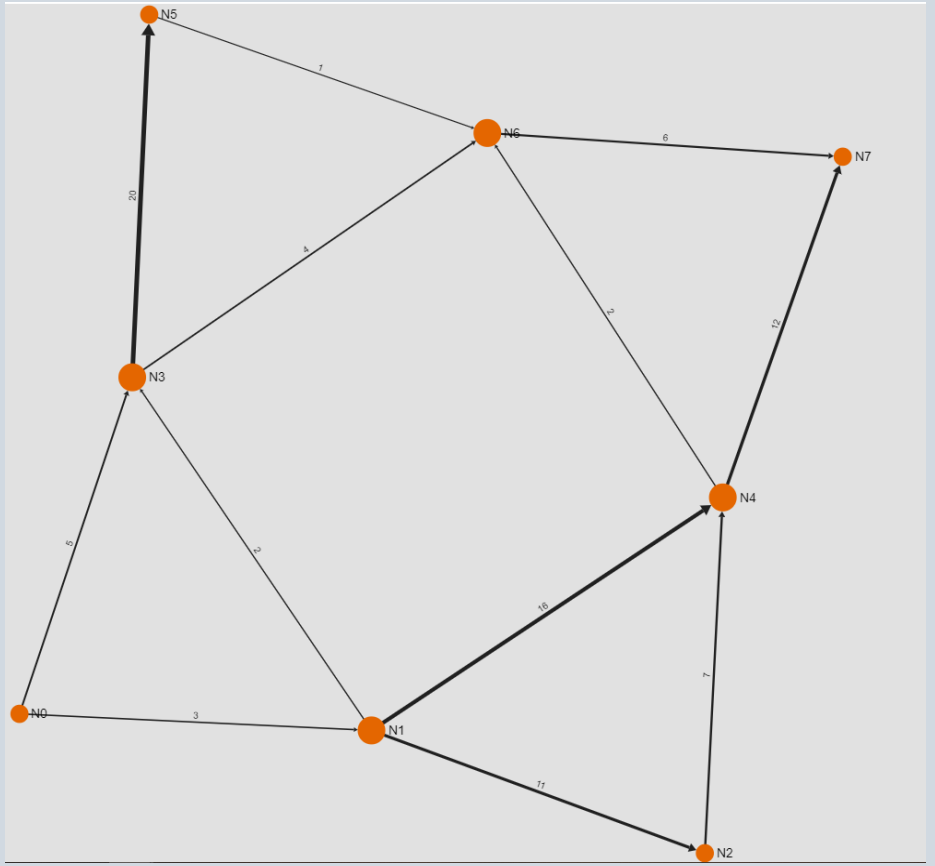
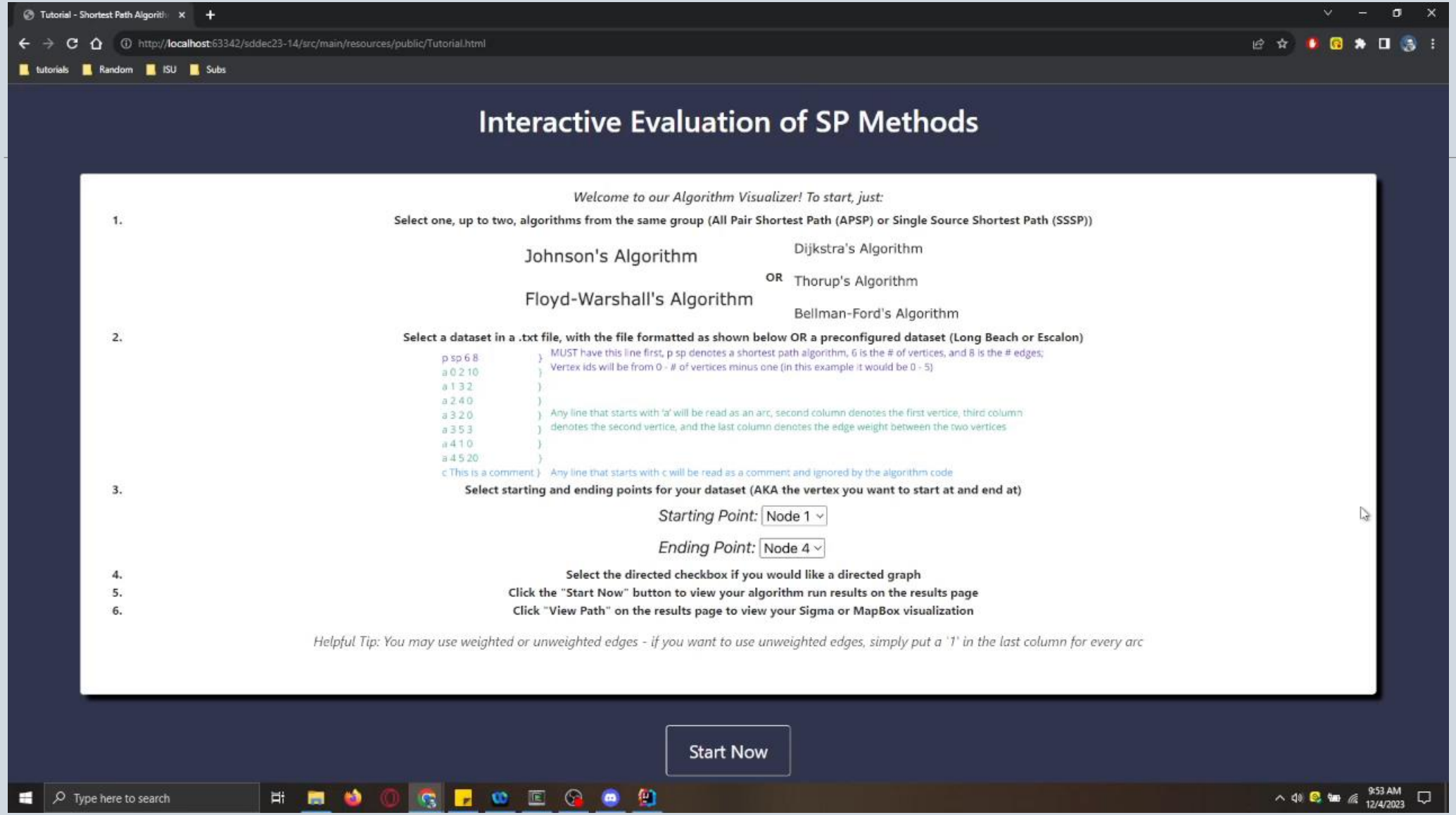


Figure 10: Example of a Sigma Visualization


Tutorial - Shortest Path Algorith
http://localhost:63342/sddec23-14/src/main/resources/public/Tutorial.html
tutorials
Random
ISU
Subs
Interactive Evaluation of SP Methods
Welcome to our Algorithm Visualizer! To start, just:
1.
Select one, up to two, algorithms from the same group (All Pair Shortest Path (APSP) or Single Source Shortest Path (SSSP))
Johnson's Algorithm
Floyd-Warshall's Algorithm
OR
Dijkstra's Algorithm
Thorup's Algorithm
Bellman-Ford's Algorithm
2.
Select a dataset in a .txt file, with the file formatted as shown below OR a preconfigured dataset (Long Beach or Escalon)
p sp 6 8
a 0 2 10
a 1 3 2
a 2 4 0
a 3 2 0
a 3 5 3
a 4 1 0
a 4 5 20
c This is a comment
MUST have this line first, p sp denotes a shortest path algorithm, 6 is the # of vertices, and 8 is the # edges;
Vertex ids will be from 0 - # of vertices minus one (in this example it would be 0 - 5)
Any line that starts with 'a' will be read as an arc, second column denotes the first vertice, third column denotes the second vertice, and the last column denotes the edge weight between the two vertices
Any line that starts with c will be read as a comment and ignored by the algorithm code
3.
Select starting and ending points for your dataset (AKA the vertex you want to start at and end at)
Starting Point: Node 1
Ending Point: Node 4
4.
Select the directed checkbox if you would like a directed graph
5.
Click the "Start Now" button to view your algorithm run results on the results page
6.
Click "View Path" on the results page to view your Sigma or MapBox visualization
Helpful Tip: You may use weighted or unweighted edges - if you want to use unweighted edges, simply put a '1' in the last column for every arc
Start Now
Type here to search
9:53 AM
12/4/2023

Testing - Frontend

User Input

UI Navigation

End-to-End Communication

Visualization Rendering

```
c an edge between node 0 and node 1 with a weight of 3
a 0 1 3
```

Figure 11: Part of an uploaded dataset file

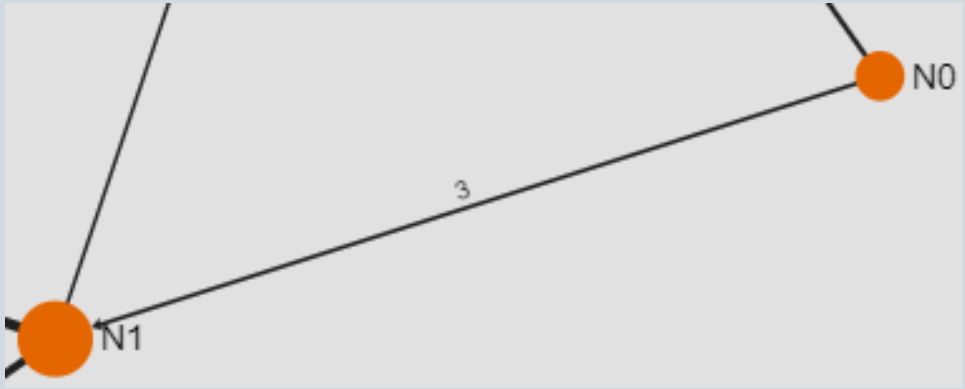


Figure 12: dataset represented in the visualizer

Testing – Backend

Testing Strategy

Test Results

133 tests; 85%+ line coverage

* Additional details are available on p. 30 of the Design Document

Lessons Learned

- Use a different programming language – C++, Rust
- Reevaluate implementation timeline

Thanks + Q&A