Interactive Evaluation of Shortest Path Methods

SDDEC23-14

ALEX BLOMQUIST
SELMA SARIC
SAMUEL CALDWELL
YADIEL JOHNSON

Problem Statement

Can you identify which one has a lower time complexity?

$O(m\ loglog\ n\ logloglog\ n)$
Stratified Binary Tree Variant of Dijkstra's

vs.

$O(m + (n\ log\ n)/(loglogn))$
Fredman and Tarjan's Fibonacci Heap Variant of Dijkstra's

Your answer is likely **No**.

- Algorithm research is always developing
- Efficiency is important, but hard to compare

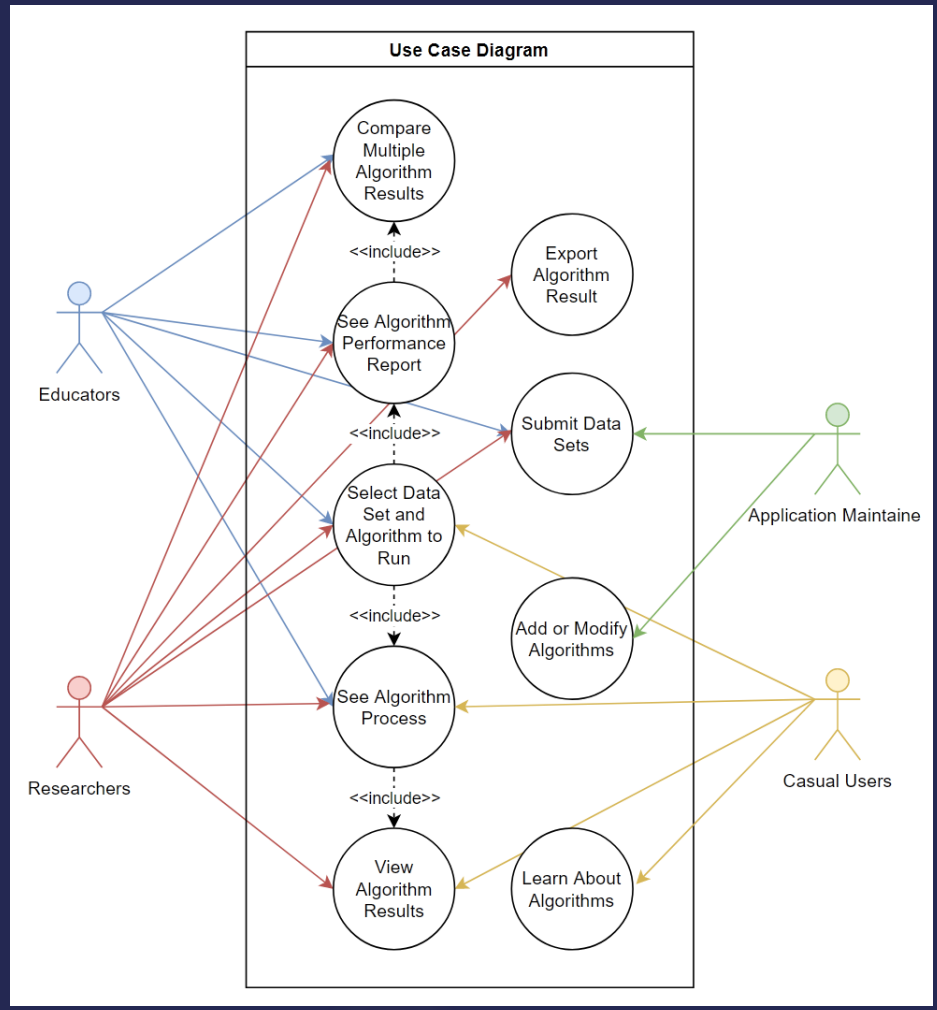
This project aims to develop a system that enables:

- The use of various algorithms
- The use of different datasets

...to output detailed comparisons

Stakeholders & Use-Cases

- **Educators**
  - Present and educate people about the efficiency of different shortest-path algorithms
- **Researchers**
  - Generate reports on the efficiency of the different algorithms
  - Compare and choose different algorithms for their own projects based on the type of data sets they work with
- **Students**
  - Tool to better understand and learn about the performance of algorithms on different data sets



Requirements & Constraints

Functional Requirements	UI Requirements
• Algorithm execution on data sets + metrics report • Visualizations of algorithm outcomes/comparisons	• User upload/algorithm selection • Clean, organized presentation of SP visualizations
Resource Requirements	**Constraints**
• Optimal resource usage per algorithm run • Report generation and storage	• Full-Stack Solution • Budget: No more than $200

Engineering Standards

- **IEEE/ISO/IEC 26514-2021**
  - Design and development of Information for users
- **IEEE/ISO/IEC 29119-1-2021**
  - Software and systems engineering – Software testing
- **IEEE/ISO/IEC 42010-2022**
  - Software, systems, and enterprise – Architecture description

Details available on page 10-11 of the design document

Task Decomposition

Frontend

- 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 metrics on the results screen.
- Develop login and account functionality

Task Decomposition

Backend

- Obtain and adapt implementations of Shortest-Path algorithms
  - If necessary, modify algorithm implementations such that all I/O operations are standardized
- Develop an “Algorithm Execution Driver” (AED)
- Develop a server component that manages RESTful transactions
  - Integrate with the driver to coordinate multiple algorithm executions
- Implement methods to receive, validate, and manage user account storage

Details available on page 16 of the design document

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)

Details available on page 18 of the design document

Project Timeline

Semester 1

Phase 1: Research and Planning		
Discover Phase | Research	2/14/23	2/14/23
TeamThink Constellation	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
SD Team Website V1	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
SD Team Website V2	4/10/23	4/23/23
Phase 3: Finishing Up		
Final Design Document	4/10/23	4/23/23
Faculty Panel Presentation	5/3/23	5/3/23

Project Timeline

Semester 2

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

Details available on page 20 and Appendix A on page 45 of the design document

Risks/Risk Mitigation Plan

Task #	Task	% Risk	Reasoning
7	Implement Driver Component	.5	Implementation Failure

Mitigation Strategy

Verify algorithm results using a variety of data sets, each with unique properties.

Details available on page 21 of the design document

Design & Broader Context

Areas Summary:

- Public Health/Safety/Welfare
  - Reduce CO2
- Global/Cultural/Social
  - Reduction in travel time
- Environmental
  - Improved code efficiency
- Economic
  - User product self-improvement

Design & Broader Context (cont.)

Prior Work/Solutions

Advantages

- Evaluate complex data sets
- Provide empirical data
- Allows direct comparison between algorithms

Disadvantages

- Less detailed visualization
- Not intended for users with no SSSP algorithm experience.

User Needs

- Educators and Students
- Researchers

More details about complexity can be found in section 4.1.4 on page 25 of the design document

Design Exploration

- **Design Decisions**
  - Web Application
  - Algorithm Scope
  - Server & Algorithm Driver
- **Decision Making & Trade-Offs**
  - HTML
  - CSS
  - Bootstrap
  - Spring Boot

INTERACTIVE EVALUATION OF SP METHODS

Register | Login

Login

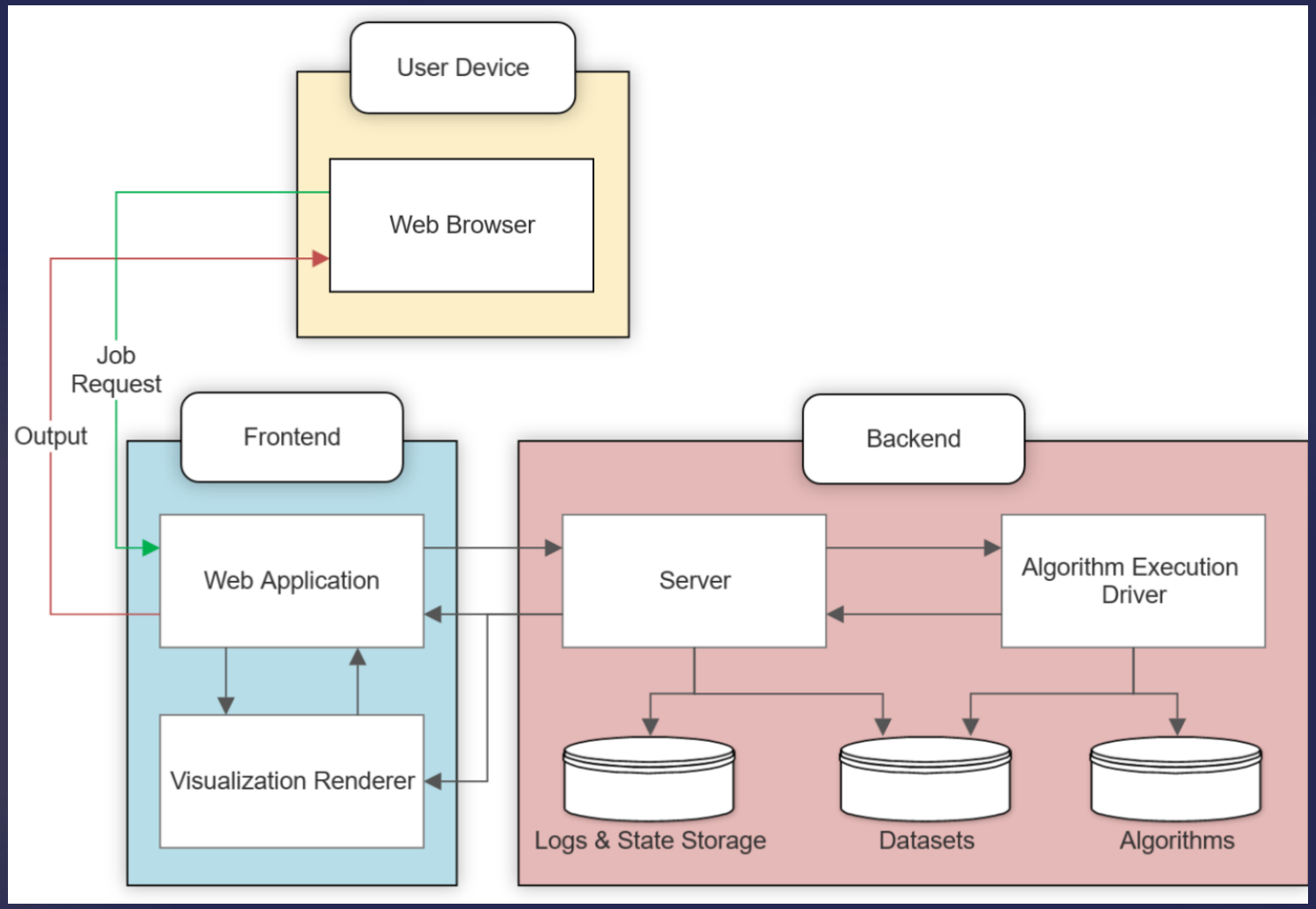
email

password

Login

No account yet? Sign up here!

Proposed Design



Details available on page 27 of the design document

Unit Testing

Frontend

- User Interface and Experience
- Login Functionality
- Visualization Renderer
  - Graphology
  - MapBox

Backend

- Algorithms
- Algorithm Execution Driver (AED)
- Logging & State Storage

Details for Unit Testing are available on page 30 of the design document

Integration Testing

Frontend

- Login Functionality
- Algorithm Selection & Upload
- Algorithm Execution Metrics
- Visualization Renderer

Backend

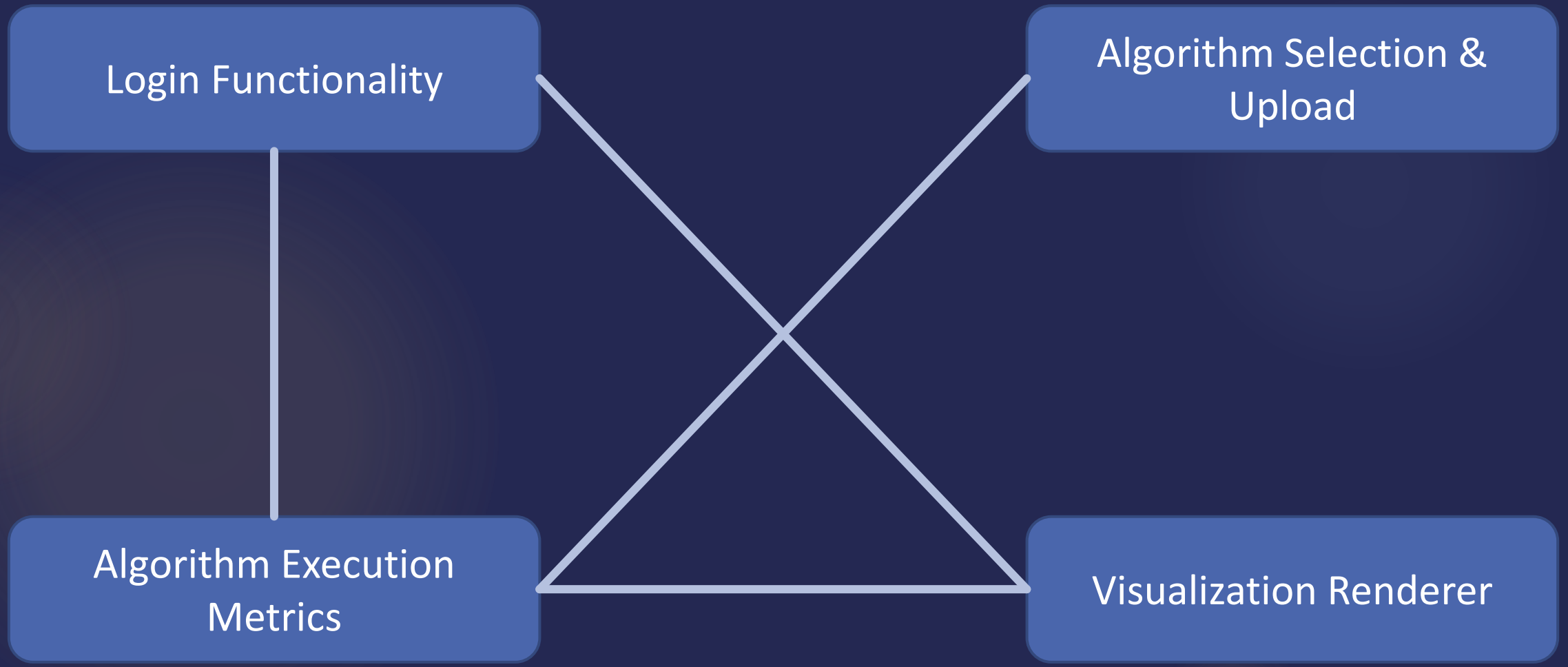
- API Endpoints
- AED Controller
- Persistence Layer

Details for Integration Testing are available on page 33 of the design document

Integration Testing

- Login Functionality
- Algorithm Selection & Upload
- Algorithm Execution Metrics
- Visualization Renderer

Integration Testing



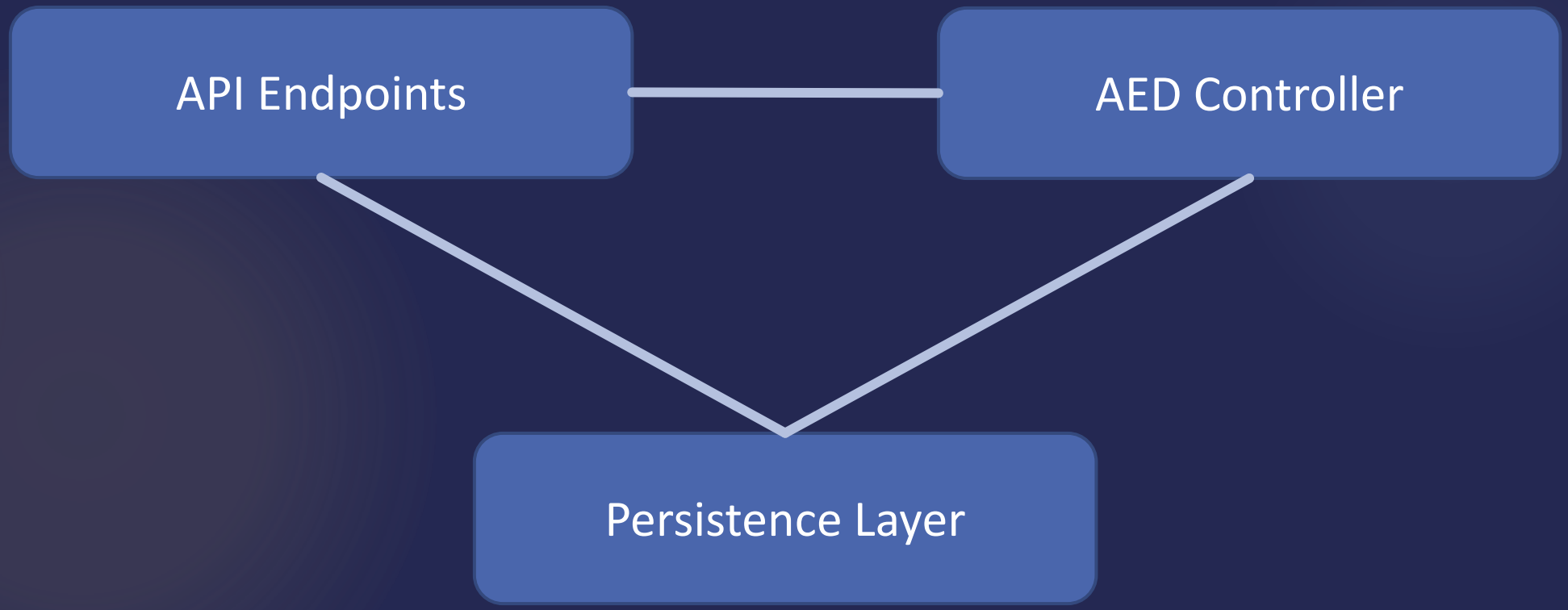
Integration Testing

API Endpoints

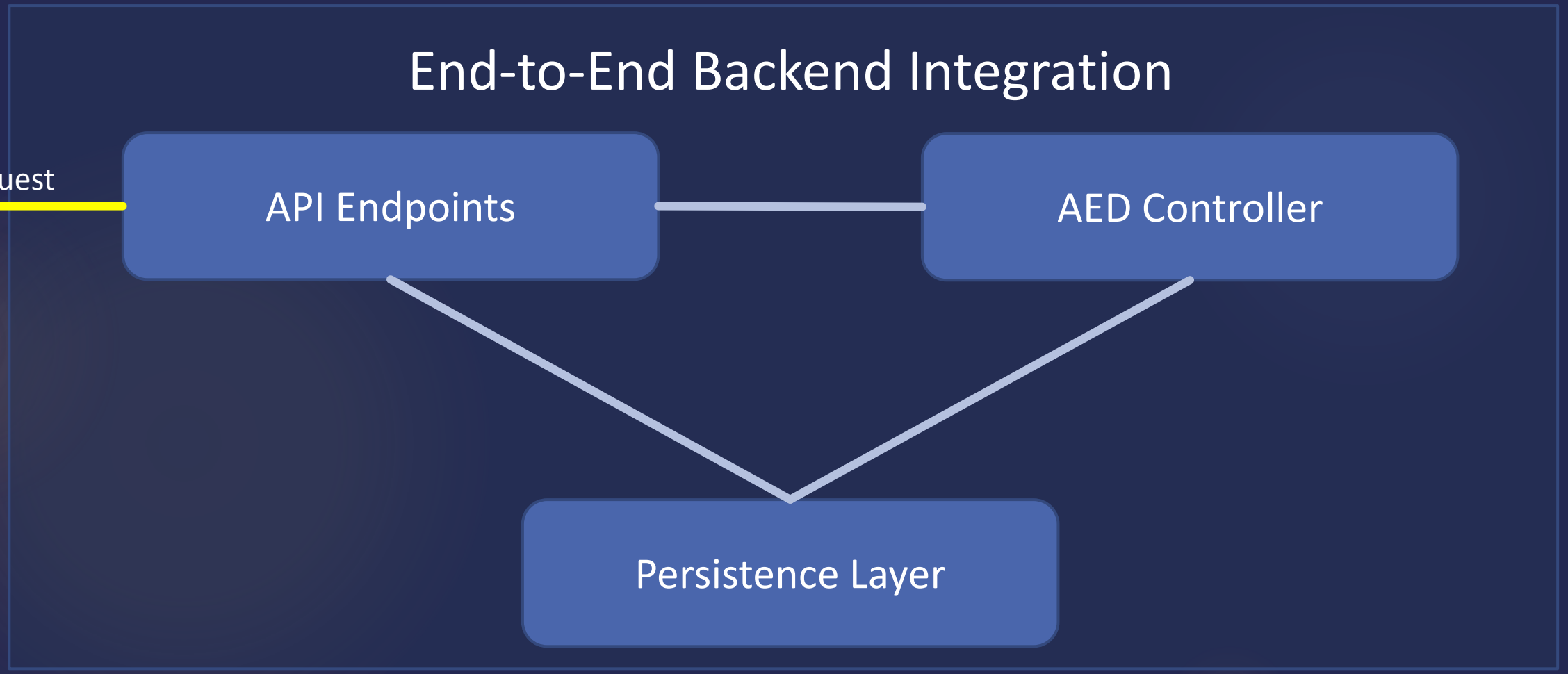
AED Controller

Persistence Layer

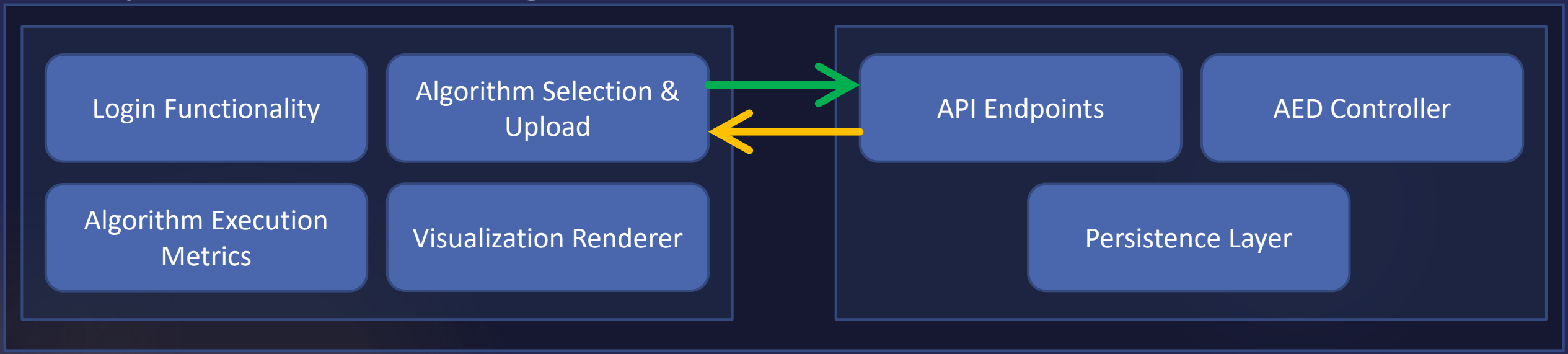
Integration Testing



Integration Testing



System Testing



System-level Testing for Functional Requirements and Use Cases Addressed				
Availability of multiple algorithms and datasets	Provide correct measurements for runtime	Distinguish algorithms based on edge weights	User can select datasets and algorithms to test	Generate informative visuals; allow for comparison & export

Details for System Testing are available on page 35 of the design document

Conclusion

Summary:

- Problem Statement
- Intended Users/Use-case
- Proposed Design
- Testing

Future Work

Summary of Implementation Tools & Objectives:

- Wireframe
- Algorithm Visualization GitHub Repositories
  - MapBox
    - Python
    - JavaScript
  - Graphology
    - JavaScript
- Shortest Path Algorithm GitHub
  - C++
  - Dijkstra's, A*, CH, etc.

Thanks + Q&A