

# CSCE 606 600 Software Engineering

## Project Report

LLNL

### Team Members (*Alphabetically*)

Brent Arnold Basiano  
Bryant McArthur  
Aubrey Moulton  
Nicholas Soliman  
Mahsa Valizadeh  
Vandna Venkata Krishnan

## Table of Contents

1. Project Summary .....	2
2. Lo-Fi Mockup .....	2
3. List of User Stories and Tasks .....	3
4. User Story Description .....	6
4.1 Add Inclusion Feature .....	6
4.2 Add Exclusion Feature .....	7
4.3 Add Budget Feature .....	8
4.4 User Login .....	9
4.5 Add Retrieve Information .....	10
4.6 Add Metadata Display .....	10
4.7 JSON File Upload Feature .....	12
5. Team role .....	12
6. Summarizing Scrum Iteration Accomplishments .....	13
7. Team Member Contribution .....	14
8. Customer Meeting Dates and Description .....	15
9. BDD/TDD Process .....	16
10. Configuration Management Approach .....	16
11. Challenges .....	17
11.1 Production Release to Heroku .....	17
11.2 Github .....	17
12. Ruby Gems .....	17
13. Repo Content and Scripts for Code Deployment .....	17
14. Links .....	18

## 1. Project Summary

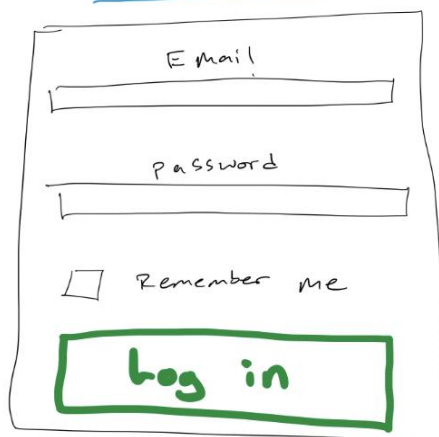
The LLNL project is to develop a dashboard to visualize knowledge graphs. Our client (stakeholder) is Professor Roger Pearce, who is a professor of practice at Texas A&M and a researcher at the Lawrence Livermore National Laboratory. Dr. Pearce has a large database of graph nodes, and has employed us to create an interactive UI to visualize the graph and allow users to explore the graph connections. The main customer use case is a user that is not an expert in the graph data, nor has knowledge of graph algorithms, but wants to explore the data for preliminary analysis.

The application is designed with two screens the user can access, and five main features. The user first interacts with a login screen where they can make an account, or be a returning user. Once the user passes the login, they can access the dashboard. The main features of the application are that a user can include or exclude a node manually, upload a json file of nodes, set a budget, and interact with nodes and edges in the graph to include or exclude them through a metadata column.

## 2. Lo-Fi Mockup

YOU NEED TO SIGN IN OR SIGN UP BEFORE CONTINUING

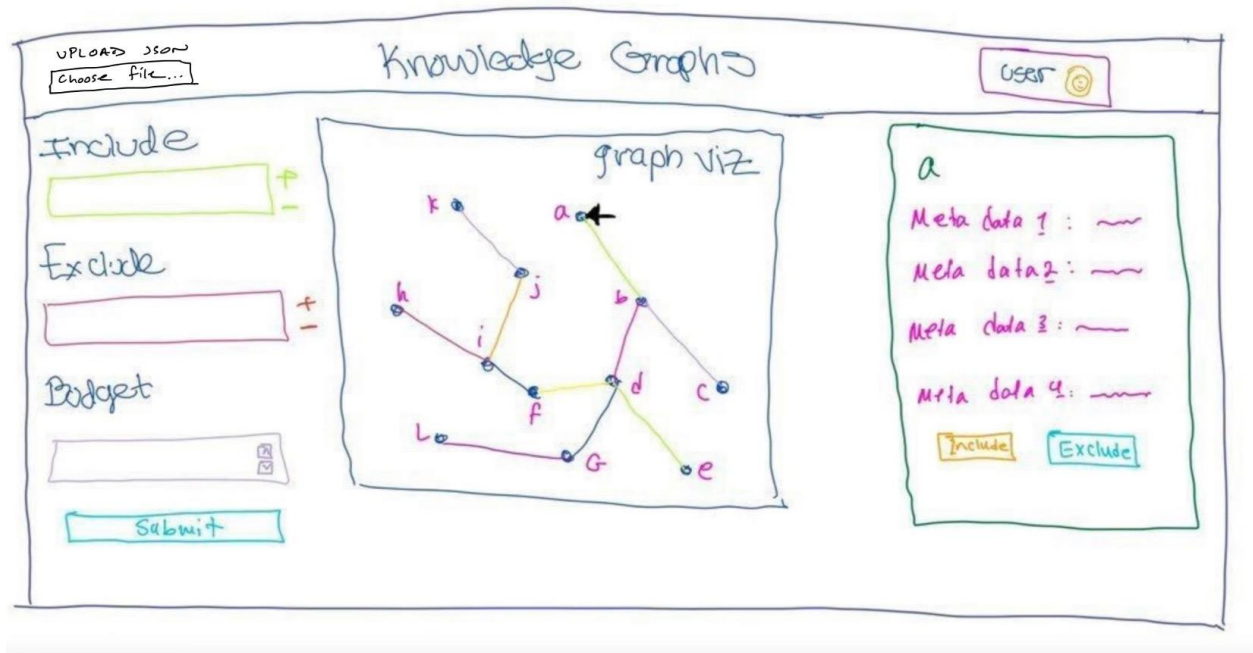
Log in



A hand-drawn login form mockup. It features two input fields labeled 'Email' and 'password'. Below the password field is a checkbox labeled 'Remember me'. At the bottom of the form is a large green rectangular button with the text 'log in' in green.

Sign up

Forgot your password?



### 3. List of User Stories and Tasks

User Stories	Status	Task points
<b>Add Inclusion Feature</b>	<b>Done</b>	<b>4</b>
<i>Create RSpec Test Cases for Inclusion</i>	<i>Done</i>	<i>1</i>
<i>"Include" Form Properly Queries Database</i>	<i>Done</i>	<i>1</i>
<i>Integrate Cytoscape</i>	<i>Done</i>	<i>1</i>
<i>Modify Inclusion Model RSpec</i>	<i>Done</i>	<i>1</i>
<i>UI refresh after form submission</i>	<i>Done</i>	
<b>Add Exclusion Feature</b>	<b>Done</b>	<b>4</b>
<i>Exclusion Model</i>	<i>Done</i>	<i>3</i>

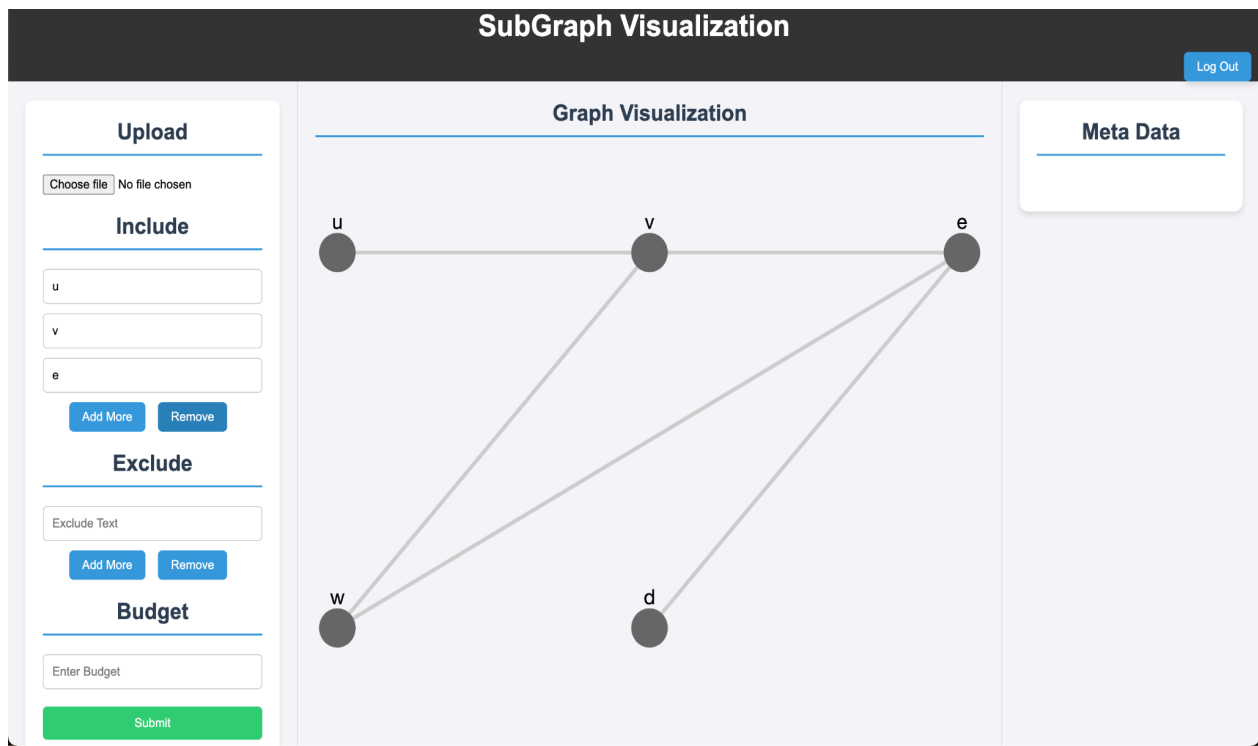
<i>Exclusion Model Rspec</i>	<i>Done</i>	
<i>Exclusion Model Cucumber</i>	<i>Done</i>	<i>1</i>
<b>Add Retrieve Information</b>	<b>Done</b>	<b>3</b>
<i>Create Frontend Form</i>	<i>Done</i>	<i>1</i>
<i>Use LLNL Algorithm (C++ working on deploy)</i>	<i>Done</i>	<i>2</i>
<b>Add Budget Feature</b>	<b>Done</b>	<b>4</b>
<i>Budget Model</i>	<i>Done</i>	<i>2</i>
<i>Budget Rspec</i>	<i>Done</i>	<i>1</i>
<i>Budget Cucumber</i>	<i>Done</i>	<i>1</i>
<b>User Login</b>	<b>Done</b>	<b>7</b>
<i>Enable a User to Login</i>	<i>Done</i>	<i>2</i>
<i>Add view for login page</i>	<i>Done</i>	<i>1</i>
<i>Add user model for user db</i>	<i>Done</i>	<i>1</i>
<i>Make design look better (both login and main pages)</i>	<i>Done</i>	<i>1</i>
<i>Cucumber tests for logging in</i>	<i>Done</i>	<i>1</i>
<i>Rspec tests for logging in</i>	<i>Done</i>	<i>1</i>
<b>Add Metadata Display</b>	<b>Done</b>	<b>7</b>
<i>Metadata button actions</i>	<i>Done</i>	<i>1</i>
<i>Integrate Cucumber</i>	<i>Done</i>	<i>1</i>

<i>Create Frontend Display for Metadata</i>	<i>Done</i>	<i>1</i>
<i>Metadata button actions Cucumber</i>	<i>Done</i>	<i>1</i>
<i>Cucumber for Metadata button actions updates</i>	<i>Done</i>	<i>1</i>
<i>Metadata button actions Rspec</i>	<i>Done</i>	<i>1</i>
<i>Metadata Button Actions</i>	<i>Done</i>	<i>1</i>
<b>JSON File Upload Feature</b>	<b>Done</b>	<b>3</b>
<i>Json upload Cucumber</i>	<i>Done</i>	<i>1</i>
<i>JSON file upload feature</i>	<i>Done</i>	<i>1</i>
<i>JSON upload Rspec</i>	<i>Done</i>	<i>1</i>
<b>General Tasks</b>	<b>Done</b>	<b>16</b>
<i>Execute c++ file locally</i>	<i>Done</i>	<i>1</i>
<i>Implement new algorithm from Roger</i>	<i>Done</i>	<i>1</i>
<i>Automation for tests</i>	<i>Done</i>	<i>1</i>
<i>Documentation for deployment</i>	<i>Done</i>	<i>1</i>
<i>Final report</i>	<i>Done</i>	<i>4</i>
<i>Demo</i>	<i>Done</i>	<i>4</i>
<i>Presentation</i>	<i>Done</i>	<i>4</i>

## 4. User Story Description

### 4.1. Add Inclusion Feature:

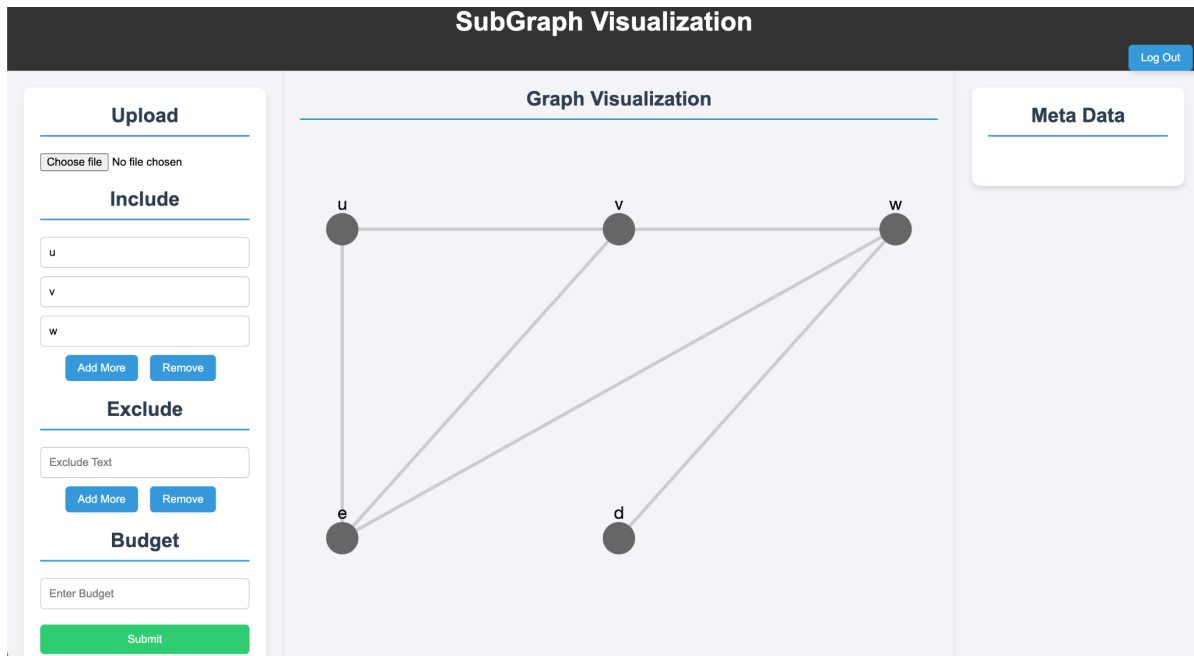
The include feature allows the users to manually input the nodes that they want to be shown in the graph. The user is given two fields to type their input, then hit submit for it to be rendered in the graph. If the user wants to add more than two nodes, they can add another field and resubmit. The inclusion story was given 4 points and it is fully implemented.



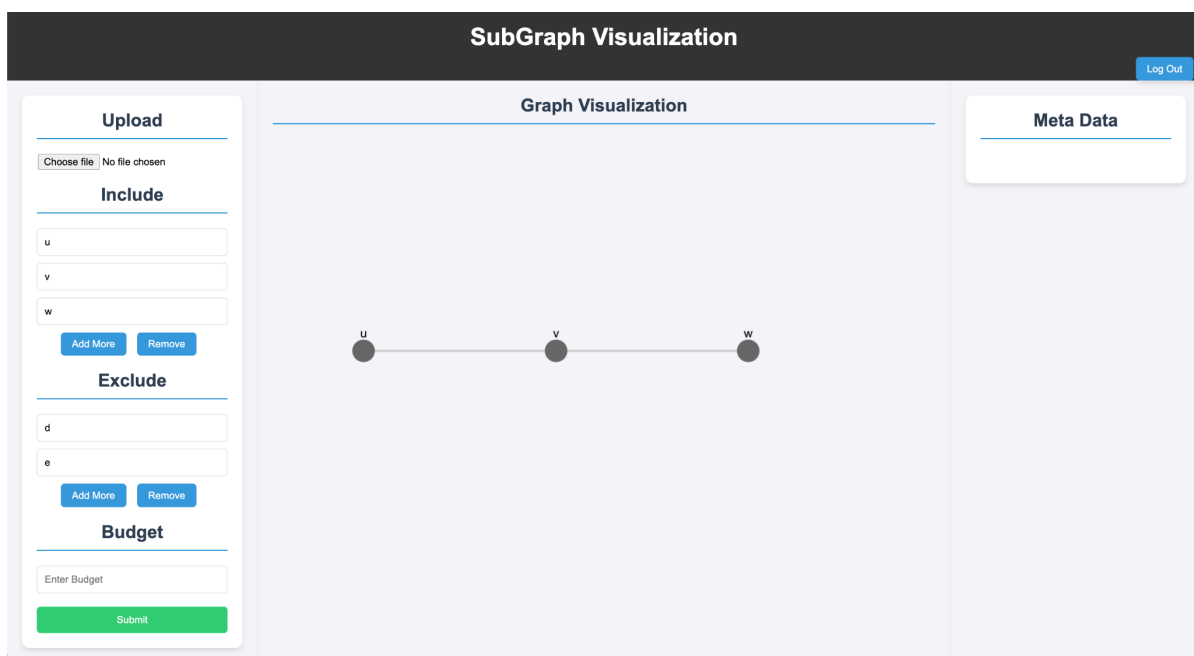
Including nodes “u”, “v”, and “e”

#### 4.2. Add Exclusion Feature:

The exclude feature enables the user to remove the nodes they do not want to display in the graph. The user is given a field to type in the node they want to exclude. The user has the option to add more excluded fields and then resubmit. The assigned point was 4 and it is fully implemented.



Including nodes “u”, “v”, and “w”

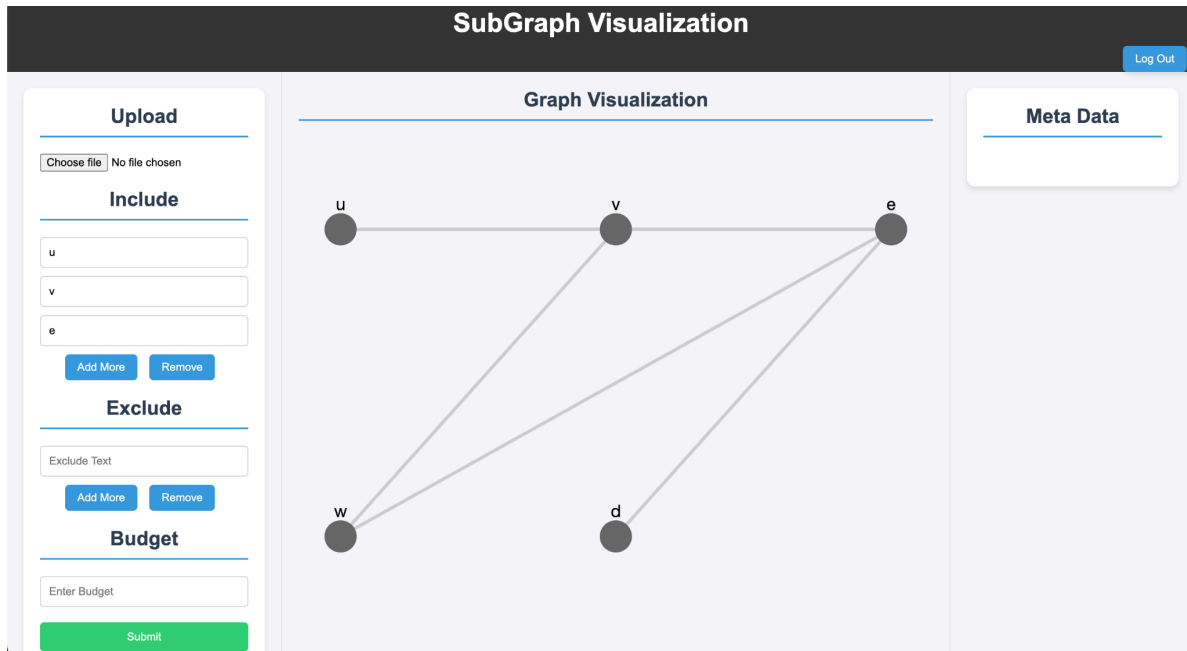


Excluding nodes “d” and “e” from displayed nodes (“u”, “v”, and “w”)

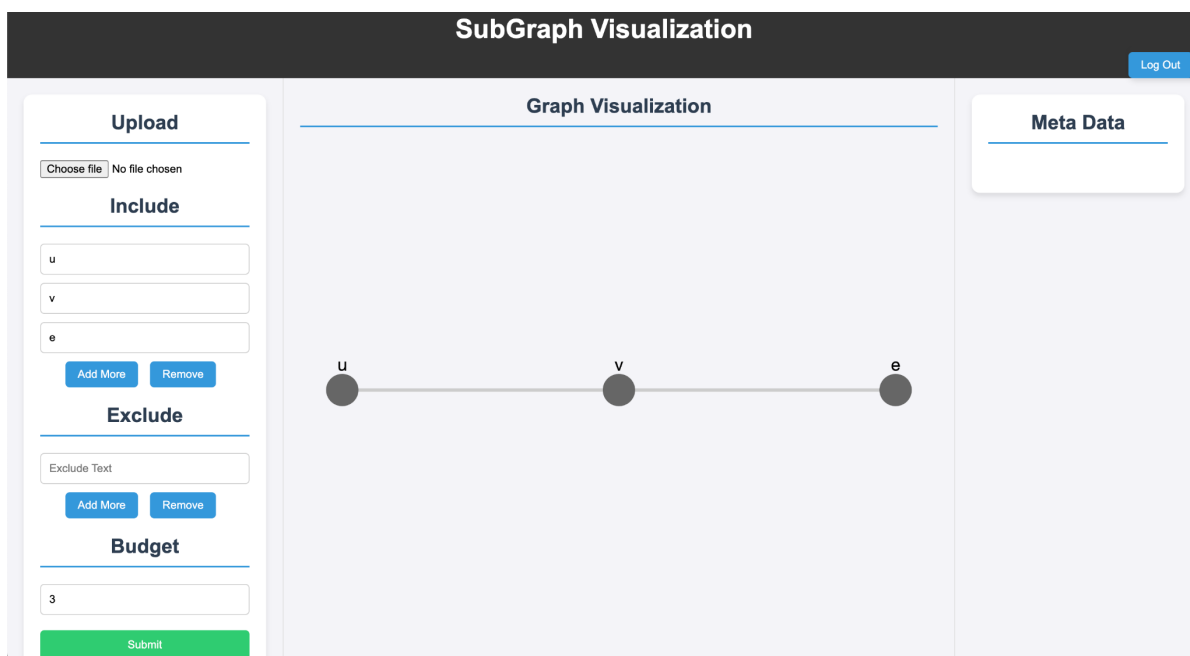


### 4.3. Add Budget Feature:

The budget feature is a limit of nodes that will be displayed in the graph that is set by the user. The user is given a field to enter a numerical value. The implementation sets a default value of 50 nodes. The minimum value allowed is 1, and the maximum is the size of the graph. The assigned point is 4 and it is fully implemented.



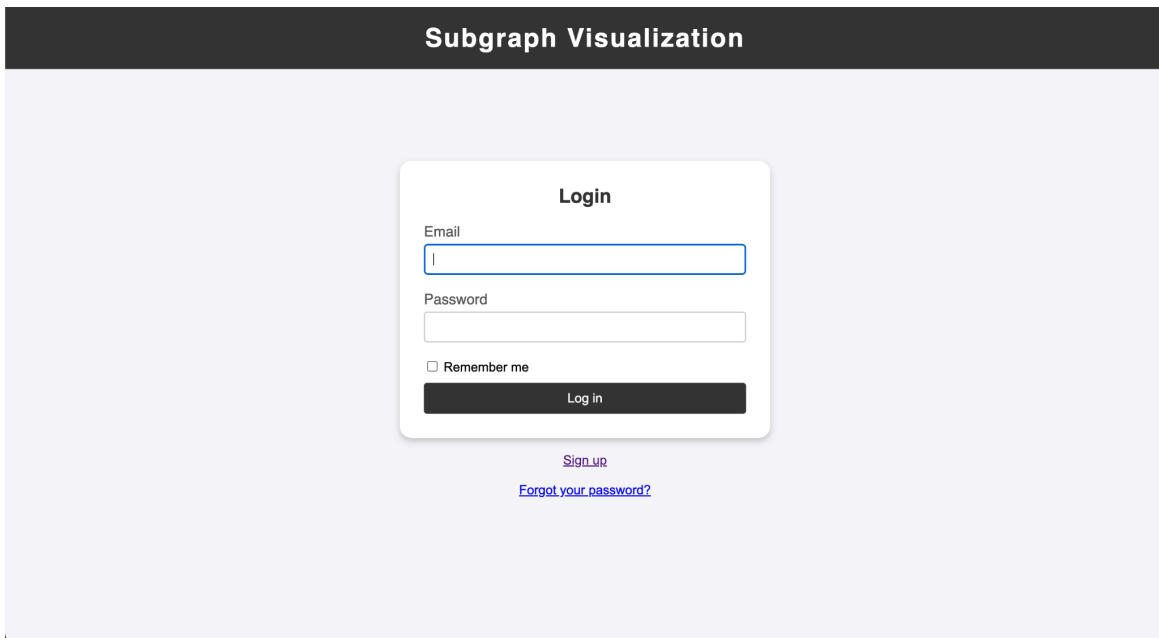
Displaying graph with budget sets to its default value, i.e., 50



Displaying graph with budget value of 3

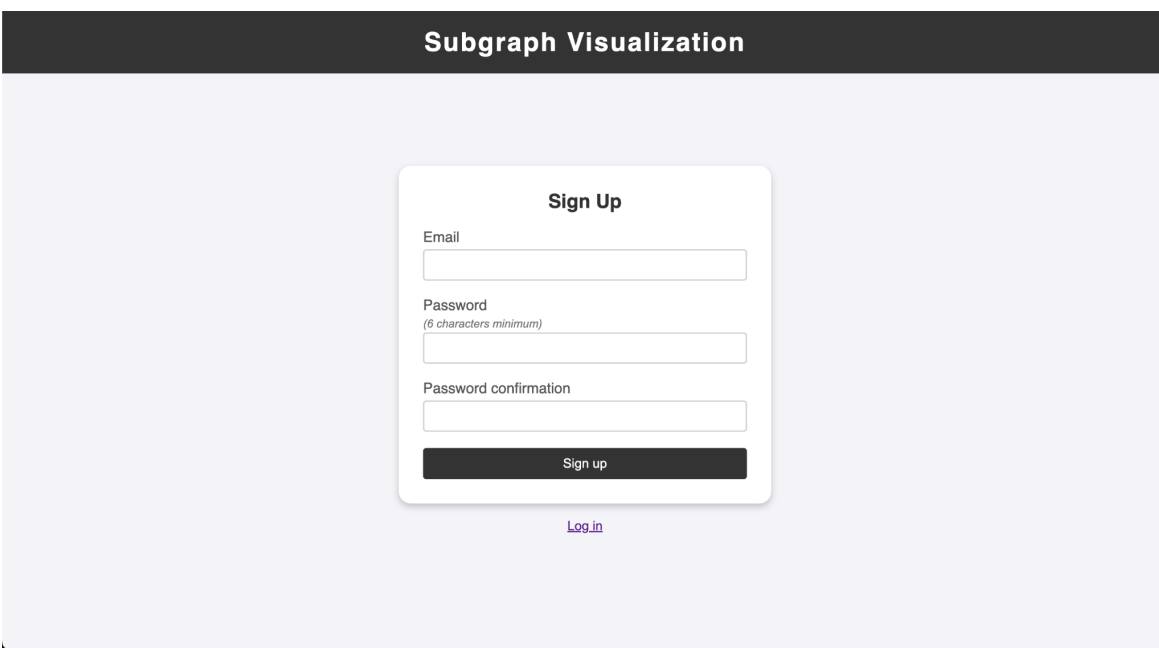
#### 4.4. User Login:

The user login feature makes the user register before being able to use the application. The first time they access the application, the user signs up with an email and password. On subsequent visits, the user can use these credentials to log back in. Also, if they forgot their password, they can restore it. The assigned point is 7 and it is fully implemented.



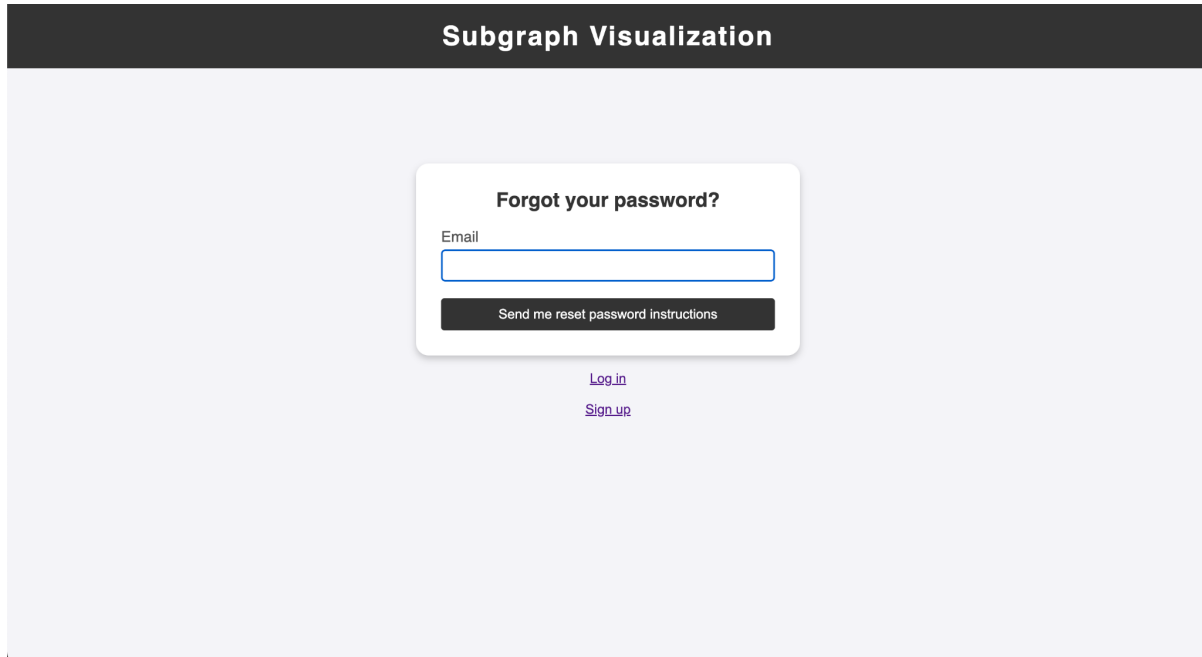
The screenshot shows a web page titled "Subgraph Visualization" in a dark header. The main content area is light gray and contains a white "Login" form. The form has two input fields: "Email" and "Password". Below the "Password" field is a checkbox labeled "Remember me". At the bottom of the form is a dark "Log in" button. Below the form are two links: "Sign up" and "Forgot your password?".

Login page



The screenshot shows a web page titled "Subgraph Visualization" in a dark header. The main content area is light gray and contains a white "Sign Up" form. The form has three input fields: "Email", "Password" (with a note "(6 characters minimum)"), and "Password confirmation". At the bottom of the form is a dark "Sign up" button. Below the form is a link: "Log in".

Sign Up page

The image shows a web interface for a 'Subgraph Visualization' application. At the top, there is a dark grey header with the text 'Subgraph Visualization' in white. Below the header, the main area has a light grey background. In the center, there is a white rounded rectangle containing a 'Forgot your password?' form. The form has a title 'Forgot your password?' in bold. Below the title is the label 'Email' followed by a text input field. Under the input field is a dark grey button with the text 'Send me reset password instructions' in white. Below the button, there are two links: 'Log in' and 'Sign up', both in a purple color.

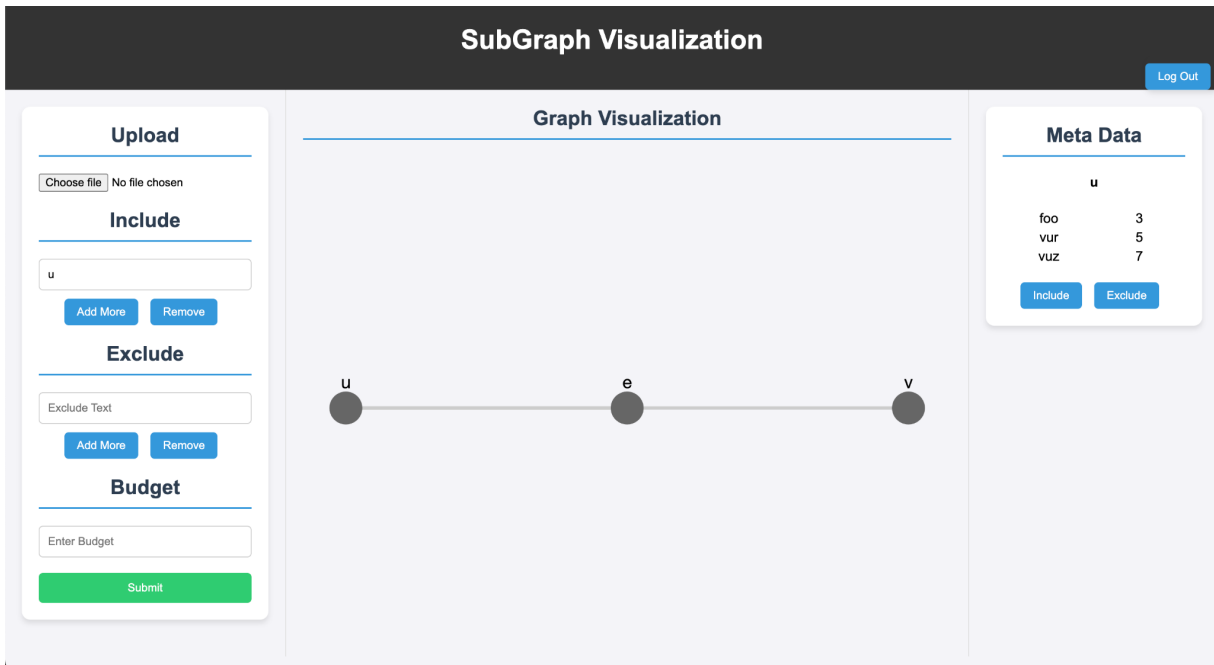
Reset password page

#### **4.5. Add Retrieve Information:**

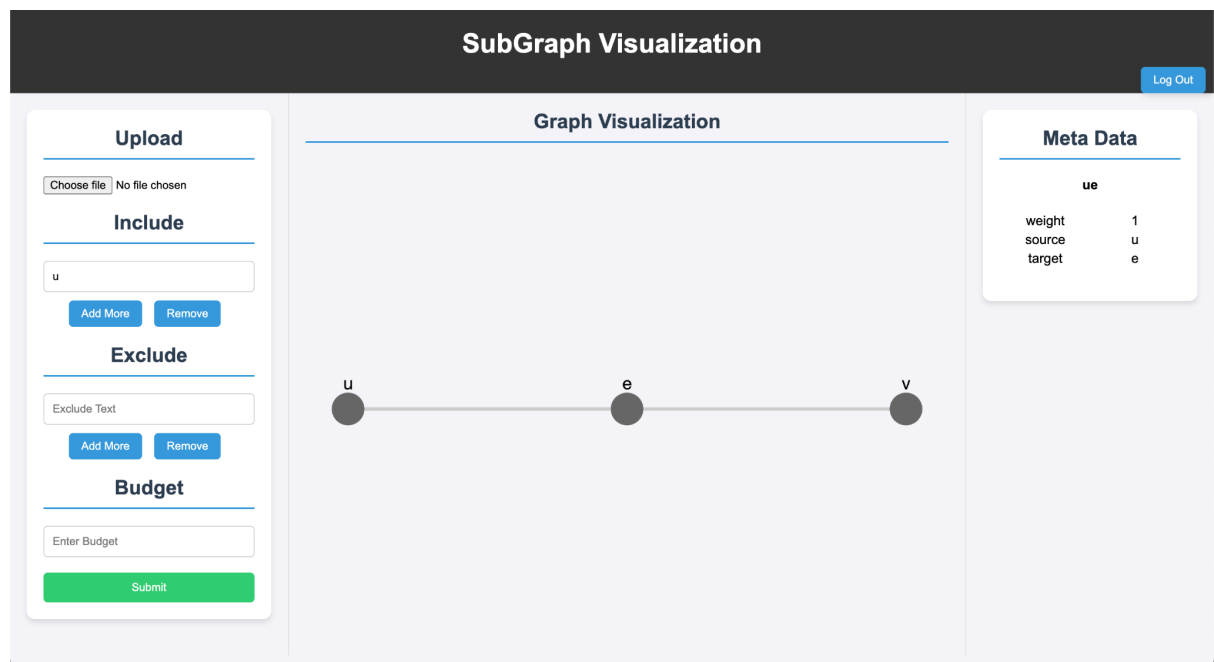
This feature, assigned a point value of 3, has been fully implemented and is designed to enhance user interaction with the UI. It enables users to efficiently input necessary information, ensuring seamless transmission of all variables and values to the backend.

#### **4.6. Add Metadata Display:**

Upon a user's interaction with a node or an edge through a click, a metadata column seamlessly appears on the right side of the interface. This column provides comprehensive details associated with the clicked edge or node.. The assigned point is 7 and it is fully implemented.



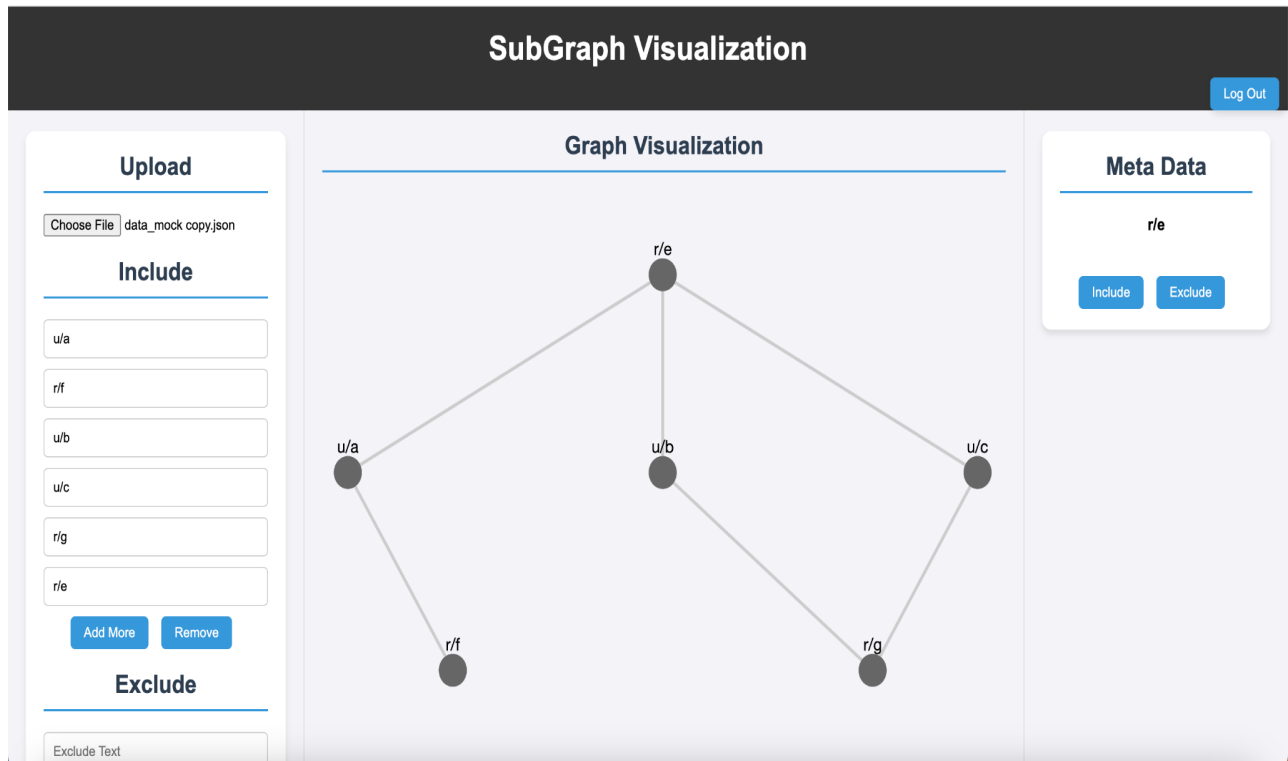
Metadata displaying node "u" information



Metadata displaying edge "ue" information

#### 4.7. JSON File Upload Feature:

The JSON File Upload Feature will allow the user to upload a json file and view the subgraph by providing the necessary input for the include, exclude and budget fields. The assigned point is 3 and it is fully implemented.



Uploading a JSON file

#### 5. Team role

- Product Owner: Bryant McArthur
- Scrum Master: Nicholas Soliman
- Developers:
  - Aubrey Moulton
  - Brent Arnold Basiano
  - Mahsa Valizadeh
  - Vandna Venkata Krishnan

## 6. Summarizing Scrum Iteration Accomplishments

For the Sprint 1, we implemented:

- Create RSpec Test Cases for Inclusion
- Integrate Cucumber
- "Include" Form Properly Queries Database
- Create Frontend Display for Metadata
- Create Frontend Form
- Integrate Cytoscape

☐ ***The completed points: 6***

For the Sprint 2, we implemented:

- Enable a User to Login
- Execute c++ file locally
- Add view for login page
- Cucumber tests for logging in
- Rspec tests for logging in
- Add user model for user db

☐ ***The completed points: 7***

For the Sprint 3, we implemented:

- Exclusion Model
- Use LLNL Algorithm (C++ working on deploy)
- Exclusion Model Rspec
- Exclusion Model Cucumber
- Modify Inclusion Model RSpec
- UI refresh after form submission

☐ ***The completed points: 7***

For the Sprint 4, we implemented:

- Budget Model
- JSON file upload feature
- Metadata button actions
- Budget Rspec
- Budget Cucumber
- Implement new algorithm from Roger
- JSON upload Rspec
- Metadata button actions Rspec
- Metadata button actions Cucumber

☐ ***The completed points: 10***

For the Sprint 5, we implemented:

- Json upload Cucumber
- Automation for tests
- Documentation for deployment
- Cucumber for Metadata button actions updates
- Metadata Button Actions
- Make design look better (both login and main pages)

☐ ***The completed points: 6***

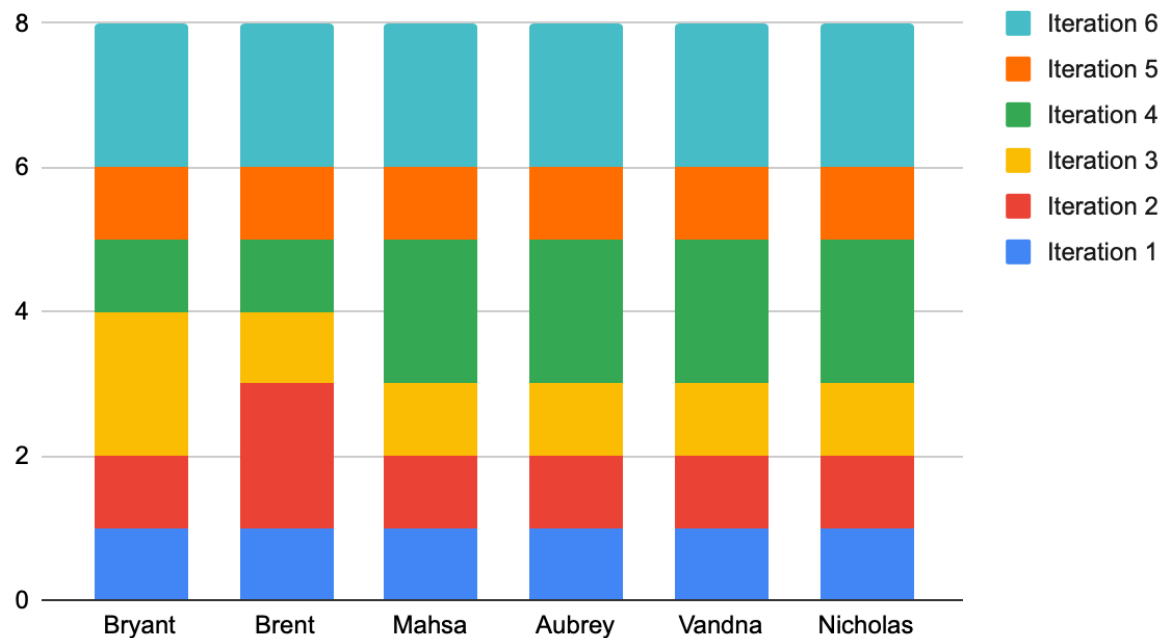
For the Sprint 6, we implemented:

- Final report
- Demo
- Presentation

☐ ***The completed points: 12***

## 7. Team Member Contribution

Team Member Performance



## **8. Customer Meeting Dates and Description**

### **8.1. September 7, 2023 at 2pm in PETR 326**

The meeting focused on exploring the concept of connection subgraphs and experimenting with their identification and visualization on the user interface. The team discussed the use of filters, including both inclusion and exclusion criteria, and the consideration of budget constraints in terms of the number of vertices to return. Experimenting cytoscape.js in visualizing connection subgraphs on the user interface.

### **8.2. September 13, 2023 at 2pm in PETR 326**

The client provided a comprehensive explanation of the use cases and the practical applications of the project. We discussed in detail the functionalities of include and exclude, as well as the format for JSON files. Additionally, the client gave us a quick summary of the existing C++ backend.

### **8.3. September 20, 2023 at 2pm in PETR 326**

Client suggested adding a metadata panel to the user interface to inspect and evaluate the metadata associated with each node. Reviewed the json format and showed a demo of the inclusion of nodes feature.

### **8.4. October 13, 2023 at 2pm in PETR 326**

Demonstrated the metadata display and exclusion feature. Further discussed the design of the login page and the authentication required for the application.

### **8.5. October 18, 2023 at 3pm in PETR 326**

The meeting covered integrating LLNL's C++ Algorithm into our application and discussed adding necessary packages and buildpacks for compiling cmake on our deployed Heroku Application.

### **8.6. October 25, 2023 at 3pm in PETR 326**

The meeting focused on selecting arbitrary budget nodes for inclusion and implementing a feature for uploading JSON files to construct new graphs.

### **8.7. November 8, 2023 at 2pm in PETR 326**

Reviewed LLNL's C++ Algorithm integration into our application. The application needs to remember the latest uploaded JSON file. The output needs to be returned in a JSON file.



#### **8.8. November 15, 2023 at 2pm in PETR 326**

Our client pointed out a need to write parameters from post to json file. As well as add a feature to display metadata for edges when an edge is clicked and changing format for upload json file for an easier read.

#### **8.9. November 20, 2023 at 1pm on Webex**

The client detailed some additional quality of life features to be added to the UI as well as recommendations for fixing CSS issues (default budget and minimum number of 'include' fields).

#### **8.10. November 27, 2023 at 2 pm in PETR 414**

Discussed with our client the final steps in concluding the project. Updated our client on the current and most likely final status of the deployed project. Informed our client of the survey that he will have to take as well as where he can access any required documentation or source code in the future.

### **9. BDD/TDD Process**

Our project used a BDD/TDD approach to ensure robust functionality and meet client requirements. This process involved writing tests before implementing features, using tools like Cucumber for BDD to define application behavior, and RSpec for TDD to ensure code quality. One significant benefit of this approach was the early detection and resolution of bugs, leading to a more reliable and maintainable codebase. However, we occasionally faced some problems in aligning test scenarios with evolving project requirements (per our customer's needs), requiring frequent updates to our test software.

### **10. Configuration Management Approach**

The approach to our project development utilizes GitHub Organization and GitHub Project to manage our project. For our branch configuration, we used a main-dev-feature branch approach where our released repo is the main branch and the dev branch is where you develop for each sprint. Any changes or additions are done by branching off of the dev branch. Dev branch is merged to the main branch at the end of the sprint. The main branch is where heroku pulls the code from.

## **11. Challenges**

### **11.1. Production Release to Heroku**

Some of the issues that were experienced during the production release process occurred when db migration was not done and the assets directory was not precompiled before release. To mitigate these problems we added a release step at the Procfile to automatically migrate the database and precompile the assets directory by running the rake command ``rake assets:precompile`` before committing and pushing to GitHub. Precompiling must be done when the ``app/assets`` directory was modified.

### **11.2. GitHub**

We did not use AWS or Cloud9, but we did use GitHub as version control for our repository. One problem we had early on was merging branches into dev or main that we thought were good, but they caused some issues when deployed to Heroku, or broke other test cases after the merge. In order to help with this we set up automated testing on Github to ensure that all our RSpec and Cucumber tests pass before merging a pull request into either the dev or main branch. This was helpful since our Heroku app directly pulled the Github repository for deployment. However, it wasn't able to prevent all potential problems. Near the very end of our project we made one merge and even though it ran perfectly locally and passed the Github test actions, when it deployed the CSS failed to load since we failed to run ``rake assets:precompile``. Our two main tools Github and Heroku overall worked well for us.

## **12. Ruby Gems**

In our project, we used a variety of additional Gems to enhance functionality. More specifically, we used SimpleCov to monitor our test coverage ensuring that we maintained high-quality code. We also used a gem called 'Rails\_12factor', which streamlined the deployment process. To ensure we implement a robust and flexible user authentication service, we used the 'Devise' gem.

## **13. Repo Contents and Scripts for Code Deployment**

The repo contents consist of the Rails app and the C++ executable for retrieving the dataset. The C++ executable called "consub" is in the public directory which the application uses to retrieve the dataset for the subgraph. The deployment follows the typical rails app deployment process as described in the repo's README .md file. For Heroku deployments, a provided Procfile is used which contains a db migration step

and the web server initialization step. All code is submitted in the GitHub repo and they are up-to-date.

## 14. Links

- Project Management Page:  
<https://github.com/orgs/LLNL-Knowledge-Graph-Exploration/projects/2/views/1>
- GitHub Repo:  
<https://github.com/LLNL-Knowledge-Graph-Exploration/LLNL>
- Deployed App:  
<https://calm-taiga-61776-a8b4b3801c6c.herokuapp.com/>
- Presentation and Demo  
[https://www.youtube.com/watch?v=2boe6MVKqoU&ab\\_channel=AubreyMoulton](https://www.youtube.com/watch?v=2boe6MVKqoU&ab_channel=AubreyMoulton)