**Title**

Augur Front End Fixes

**Description**

In this project, we plan to make the Augur interface more intuitive by including better graph visualization capabilities. There may be multiple different ways to visualize the same relationships between data and it must be clear enough for the user to understand. Augur takes data from open-source software and analyzes it to extract data about multiple different parts of the project, such as number of lines of code changed, number of pull requests, etc. and displays it on the frontend using graphs. Because there may be multiple different ways to measure good code, the user must be able to see the measurements clearly using graphs created for specific purposes.

Inside the coding features, the users’ objective is to pass their data into the software prototype that is implemented from the open-source communities. The visualizations and results extracted from the users’ resources must satisfy the following criteria: the complexity of the code, the feedback cycle and evaluations from different pull requests, and the metric for tested units. These criteria will be measured using data displayed on the graphs.

**Triggers**

Organizations have been a driven source for open-source communities, as their products utilize some open-source products for scalability projects in the industry. With that, understanding the quality of software and reproducibility would allow organizations to utilize and collaborate to either innovate new projects or enhancing existing projects. As visualizations are the best tools to understand different trajectories for those tools, creating a good front-end visualization would give more engaging discussions for better directions.

**Actors**

Stakeholders, users, communities that reproduce the code for their own purposes

**Preconditions**

Users need to be able to visualize the data’s reproducibility and coverage.

**Main Success Scenario (Goals)**

Code should be made as simple as possible. Some constant criteria should be usable across multiple different projects. Contributions to the project should fall into guidelines made by the community and industry.

**Alternate Success Scenarios**

The user is able to visualize and compare the data by searching for it. The user is able to compare similar projects using graphs to visualize differences.

**Failed End Condition**

The user is unable to understand what the data given means for their project. The user is unable to implement fixes for weak parts of their project given the visualization.

**Extensions**

Github platform and commands to rapidly test the code quality

Essential extension of VSCode to elaborate in the coding schemes.

**Steps of Execution (Requirements)**

A user should be able to provide their needs and requirements within the code complexities

A user could understand the metric of visualization of the graphs and database processing for analysis and SQL commands usage.

A user could rapidly change based on different requests of contributions and applications for their own code.

**A use case diagram**

**Diagram

Description automatically generated**