**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: display data about the project that shows the viewer the relationship between users’ activities and timelines, specifically people who have changed the project and where the changes were made. Also we include the visualization of how the project’s activities are changed overtime. With that, we make sure the visualizations could reflect the relationship among numbers of commits, repositories, pull requests, and users.

**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 enhance existing projects. As visualizations are the best tools to understand different trajectories for those tools, creating a clear front-end visualization would give more engaging discussions for better directions. Specifically, we want to utilize different types of charts and graphs, such as pie charts, line graphs, or bar graphs, that align with specific questions being addressed in the graphs. Users could then have different choices of graphs to understand the subject of their interest and to intuitively seek answers for their desired concerns from those graphs.

**Actors**

* Product and business managers.
* Professors and students.
* Augur contributors and followers.
* Linux Foundation.

**Preconditions**

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

**Main Success Scenario (Goals)**

Visualizations of the project should fall into guidelines and should allow for easy decision making by the community and industry.

Visualizations should prompt more ideas for contributors and academics to explore different relationships between users and open source data.

Visualizations should reflect the health of open source activities (either the quality is getting better, worse, or neutral in defined metrics) from interactions with different users.

Visualizations should be reusable to provide guidelines for managing and monitoring open source activities.

**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 command usage.

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

**A use case diagram**

