# Sprint 3 Progress Report

## GitHub Repository

[Provide link to GitHub repository, which should contain:

1. README.md file that explains how to deploy your code (if you add/change things from usual, otherwise this is probably a copy of Augur's README.md)
2. Progress Report, which will be some kind of indication about whether or not you need to update your scope.

]

GitHub Repository Link: <https://github.com/JacksonHaskamp/augur>

Branch for Sprint 3: sprint3 <https://github.com/JacksonHaskamp/augur/tree/sprint3>

## Scope

[Indicate whether or not the project scope needs to be updated.]

* For sprint 2, we were focused on creating visualizations based off of json data Dr.Goggins provided us. Now we need to shift towards looking at python endpoints and how they work in augur.
* Need to link augur-endpoints with our google charts visualizations
* For gathering JSON records and data in future sprints we will utilize an API endpoint from augur that will supply the data.

## Updates to Design Document and Requirements

[Indicate changes made to design document and requirements, if we made them.]

* Little to no changes to design document/requirements

## Testing Plan

[What is our testing plan? May depend on project context.]

* If we get the Python endpoint working, we could try to test it with different data.
* Pytest unit testing could be used to test the Python code, but we would need to have Python code written for this to be useful.
* For the visualizations, they could be checked to see if they make sense? For example, we could check if the axes are labeled correctly.
  + We could ask someone unrelated to the project to see if they can identify pertinent features of the data.

## Team Reflection

[For this sprint, explanation of obstacles encountered, reflections, and goals.]

* Having trouble understanding how our data will be supplied and stored
  + Will it be through json supplied to us? Or do we need to query a database?
  + ANSWER: We found that JSON data will be supplied to us via another API endpoint
* Proceed with plan to modify code to fit our data / needs
  + Figure out how to integrate / meld google charts with the augur code
* We could fetch upstream using the GitHub web interface or using CLI.   
  <https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/working-with-forks/syncing-a-fork>
  + “Configuring a remote for a fork” <https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/working-with-forks/configuring-a-remote-for-a-fork>
    - git remote -v
    - git remote add upstream <https://github.com/chaoss/augur.git>
    - git merge upstream/main
  + git-fetch <https://git-scm.com/docs/git-fetch>
* For Sprint 3, we weren’t sure what the wireframes for our project would look like. We weren't sure since our project is about making an API endpoint to produce visualizations. There didn't seem to be much UI component to our project to mock up with wireframes.
  + This gave us a vision for our final product
  + We made a wireframe depicting the Augur API documentation website and a visualization.
* We weren't sure what was meant by "cluster," although it seemed to have to do with a machine learning worker related to computational linguistic analysis, according to this GitHub issue related to clustering <https://github.com/chaoss/augur/issues/1319>
  + Dr. Goggins gave us more info about the clusters visualization: We can ignore the “max” field. Using most recent data which is pulled about every 30 days.
* For visualizing the clusters.json data, we were also having technical issues with the data types for Google Charts since dates are a different data type than numbers. We weren't sure how to visualize the data/which type of chart would be appropriate since we didn't have a clear idea of what it represented.
* Notes about Python API (Andrew Brain gave Erika some pointers)
  + create\_routes
  + Can use existing files in Augur repository as a template. Copy file and remove what you don’t need. E.g. augur/routes/nonstandard\_metrics.py file
  + Don’t need metrics
  + pull\_request\_reports.py has example query, and we can use this code   
    @server.app.route('/{}/pull\_request\_reports/average\_commits\_per\_PR/'.format(server.api\_version), methods=["GET"])
  + server.augur\_app.database is a SQLAlchemy database object (a connection)
  + Can write query using valid SQL