4320 - Final Project

Cyclones

Table of Contents:

CS 4320 - Software Engineering 1
Dr. Sean Goggins

Sprint 1: Sunday, April 10th, 2022
Sprint 2: Sunday, April 18th, 2022

Sprint 1

Important Links:

To access server:

- http://ec2-54-174-126-193.compute-1.amazonaws.com

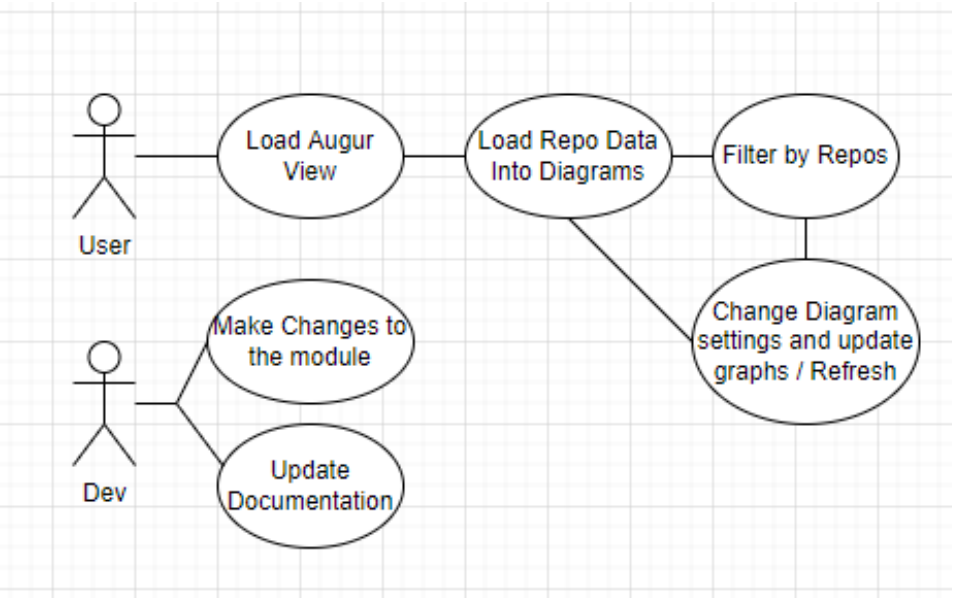
To access Github Fork:

- https://github.com/BEBeach/augur/tree/BEBeach-Sprint-1

Basic Design:

Calendar:

- Based off of the Augur View project
- Main Goals:
    - Show Calendar view
        - Each shows the following data
            - Repo Name and data
            - Time Repo was accessed
            - Breakdown of server load
        - Keeps track of historical trends
            - Ratio_abs < 1 from issues
    - Show list of late repos
        - Repos that have not been checked in x time
    - Get statistics of server load
        - Provide data to detect trends in server usage
- Stretch Goals:
    - Create user profiles / Store data to be seen over time
- Use Case Diagram:



Attributes:

PullRequest:

- Repo_id: int
- Repo_name: String
- Ratio_abs: Double
- Last collected: Date

Repos:

- Repo_id: int
- Repo_name: string
- Commits_all_time: int
- Issues_all_time: int
- Repos_status: string
- Url: string
- Color: string (not from json Data)

Requirements

User	**Activity**	**Associated Data**	**Description**
User	View Graphs made by Augur View	Module, Json data	The user can view specific data in the augur view.
Dev	Update documentation	Module	Can update the documentation of the module
User	Select Repo to	Module, Json data	The user can select a repository on

	view		the augur view.

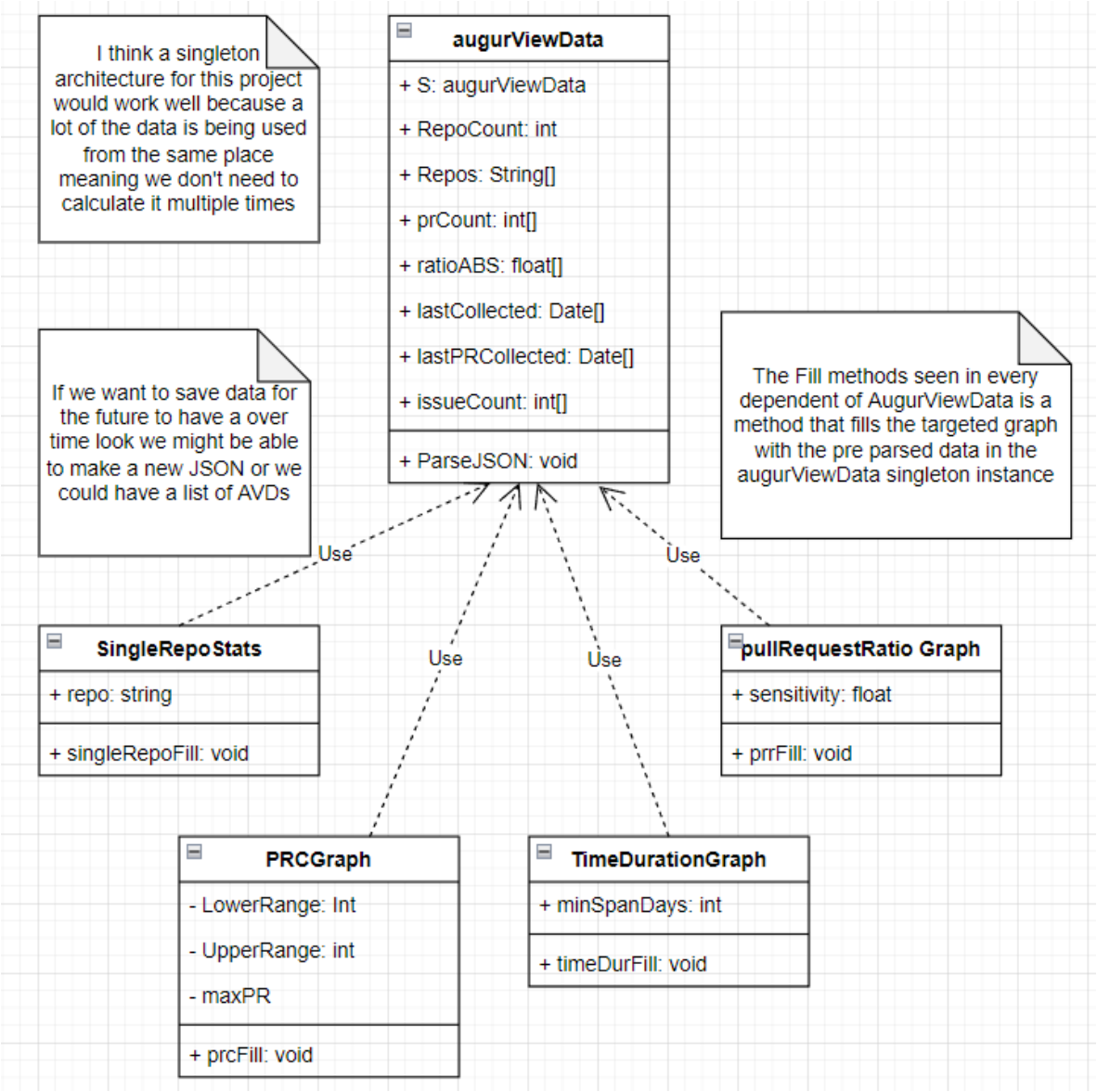
Sprint 2

Reworked Requirements:

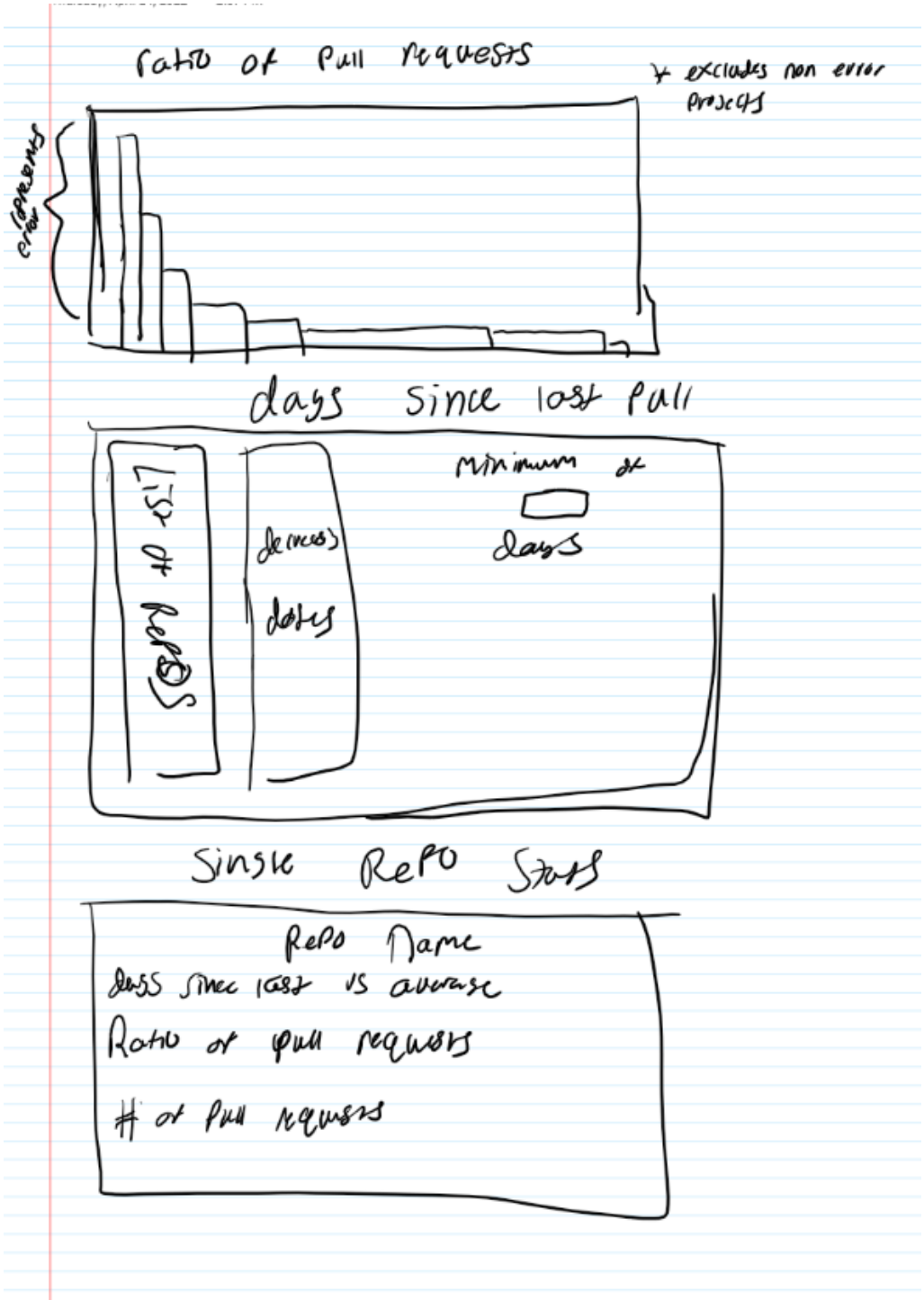
During this sprint, we decided to continue similarly to our original requirements; however, instead of making a calendar, we decided to show the data after the software has been run once. We will report any new issues that came up during the run, and we will still show the length of time that has passed between the run in question and its predecessor. This data will be shown for each repository.

Design Documents:

Data Structure

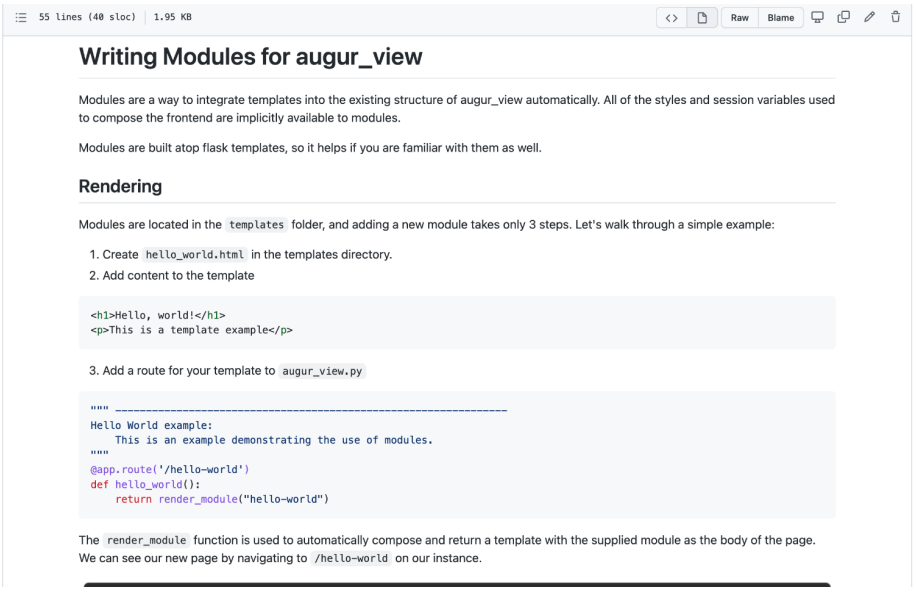


New UI look (very rough draft of what we are going for)



Hello World Example:

This example demonstrates a locally hosted version of Augur View that has been modified. A live server implementation that pulls from the official Augur system will be implemented with help from the TA.



55 lines (40 sloc) | 1.95 KB

Raw Blame

Writing Modules for augur_view

Modules are a way to integrate templates into the existing structure of augur_view automatically. All of the styles and session variables used to compose the frontend are implicitly available to modules.

Modules are built atop flask templates, so it helps if you are familiar with them as well.

Rendering

Modules are located in the `templates` folder, and adding a new module takes only 3 steps. Let's walk through a simple example:

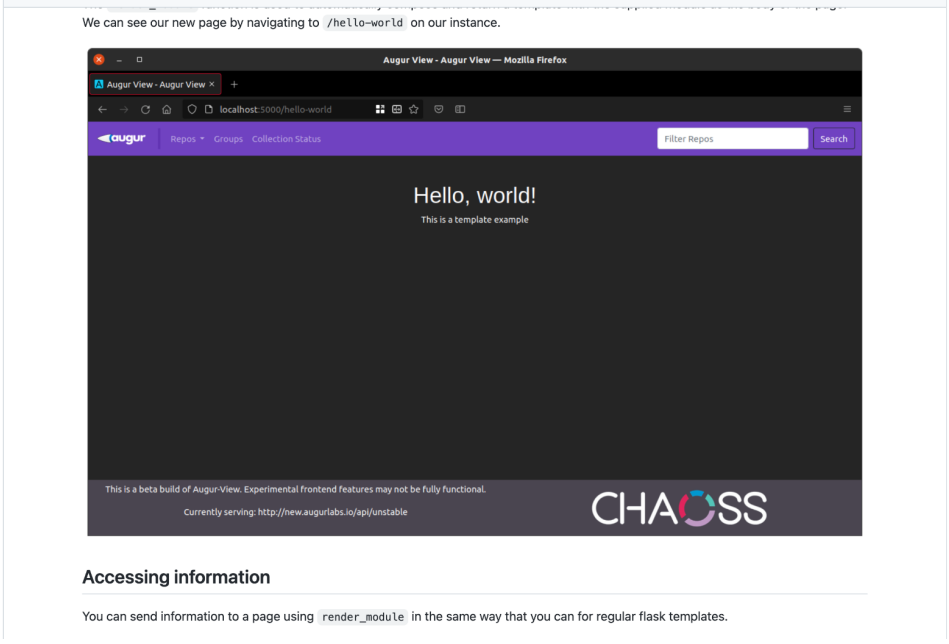
1. Create `hello_world.html` in the templates directory.
2. Add content to the template

```
<h1>Hello, world!</h1>
<p>This is a template example</p>
```

3. Add a route for your template to `augur_view.py`

```
""" ----------------------------------------------------------------
Hello World example:
    This is an example demonstrating the use of modules.
"""
@app.route('/hello-world')
def hello_world():
    return render_module("hello-world")
```

The `render_module` function is used to automatically compose and return a template with the supplied module as the body of the page. We can see our new page by navigating to `/hello-world` on our instance.



We can see our new page by navigating to `/hello-world` on our instance.

Accessing information

You can send information to a page using `render_module` in the same way that you can for regular flask templates.

To-do

- Name - Priority - Developer
- Launch Augur/augur view on remote server - 1 - Brian
    - Augur View Hello World
- General Graph Format
    - Inverse order, greatest -> least
    - Logarithmic view
- Pull Request Count Graph - 3
    - List where pull requests != pull completed
    - Possibly Remove add to the single repos
    - General Graph Format
- Missing Pull Requests Graph - 1
    - Combine with above graph
    - General Graph Format
- Pull Request Ratio - 1 - Quinton Thuet
    - Remove all ones that have are >= 1
    - General Graph Format
- Issues - 4
    - General graph update
- TimeLine - 2
    - Complete Rehaul to make it more readable
    - Find greatest duration and list it
    - Click to expand list
    - General Graph Format
- Commit Completion - 5
    - List incomplete collections
    - State if there are none
- Single Repo Status - 5 [stretch goal]
    - New Metrics to look at compared to averages

Tools and Links

- Github Forks
  - Augur View Fork
    - https://github.com/BEBeach/augur_view
  - Augur Fork
    - https://github.com/BEBeach/augur
- EC2 Instance
  - Make sure the server is running
  - To connect to server via browser
    - ec2-44-201-91-139.compute-1.amazonaws.com
  - To connect to server via terminal:
    - Contact Brian Beach at bebz4t@umsystem.edu
    - Download the pem key
    - Run this terminal command with the complete file path for the .pem file
    - ssh -i "4320-Server-1.pem" ubuntu@ec2-44-201-91-139.compute-1.amazonaws.com
    - Should look like this;
      - ssh -i /Users/brian/Programming/4320/4320-Server-1.pem ubuntu@ec2-44-201-91-139.compute-1.amazonaws.com