SI 507: FINAL PROJECT DATA CHECKPOINT

VISUALIZATION OF REPOSITORIES ON GITHUB

NAME: ADITYA MENON

UNIQNAME: adityame

Link to project code: https://github.com/supraditya/SI507_final_project

Note: The main application is within app.py. Just clone the repository, install the required modules, and run app.py through flask. The files 'final_proj.ipynb' and 'final_proj.py' are non-essential and are not needed for program execution.

Data sources: A user's GitHub repository data (both public and private)

For example: https://github.com/supraditya/SI539Waiver-2022

Data source used for testing: https://github.com/supraditya

What is the type of data that you're working with?

The response is parsed as JSON before being used.

How was the data accessed?

The data has been accessed via **OAuth** through Github's REST API.

The accessed data (at the time of writing) consists of the list of all branches within a selected repository, and the entire commit history of repositories (both public and private)

Caching

Caching has been implemented through python and local JSON file. The application checks the cache for relevant keys (and whether the data within the cache is the same as the recently fetched data) before updating the cache with the fetched data.

Evidence of caching can be found within the cache.json file on Github.

Additionally, this is a sample of the type of content that cache.json contains after some test executions:

{"account_info": {"login": "supraditya", "id": 43109943, "node_id": "MDQ6VXNlcjQzMTA5OTQz", "avatar_url": "https://avatars.githubusercontent.com/u/43109943?v=4", "gravatar_id": "", "url": "https://api.github.com/users/supraditya", "html_url": "https://github.com/supraditya", "followers_url": "https://api.github.com/users/supraditya/followers", "following_url": "https://api.github.com/users/supraditya/following{/other_user}", "gists_url": "https://api.github.com/users/supraditya/gists{/gist_id}", "starred_url": "https://api.github.com/users/supraditya/starred{/owner}{/repo}", "subscriptions_url": "https://api.github.com/users/supraditya/subscriptions", "organizations_url": "https://api.github.com/users/supraditya/orgs", "repos_url":

"https://api.github.com/users/supraditya/repos", "events_url": "https://api.github.com/users/supraditya/events{/privacy}", "received_events_url": "https://api.github.com/users/supraditya/received_events", "type": "User", "site_admin": false, "name": "Aditya Menon", "company": null, "blog": "www.supraditya.com", "location": null, "email": null, "hireable": true, "bio": "\r\n   Front-end Web Developer | UI/UX Designer\r\n", "twitter_username": null, "public_repos": 28, "public_gists": 0, "followers": 16, "following": 16, "created_at": "2018-09-09T10:42:50Z", "updated_at": "2022-12-05T01:29:56Z"}, "repo_name": "SI539Waiver-2022", "branch_data": [{"name": "js-comp", "commit": {"sha": "dce1edfadd8572fe67c8b1e1d0eedb92545f6d09", "url": "https://api.github.com/repos/supraditya/SI539Waiver-2022/commits/dce1edfadd8572fe67c8b1e1d0eedb92545f6d09"}, "protected": false, "protection": {"enabled": false, "required_status_checks": {"enforcement_level": "off", "contexts": [], "checks": []}}, "protection_url": "https://api.github.com/repos/supraditya/SI539Waiver-2022/branches/js-comp/protection"}, {"name": "main", "commit": {"sha": "7f618726d2cd354e8c0ff758406fa0b6e648d98f", "url": "https://api.github.com/repos/supraditya/SI539Waiver-2022/commits/7f618726d2cd354e8c0ff758406fa0b6e648d98f"}, "protected": false, "protection": {"enabled": false, "required_status_checks": {"enforcement_level": "off", "contexts": [], "checks": []}}, "protection_url": "https://api.github.com/repos/supraditya/SI539Waiver-2022/branches/main/protection"}]}

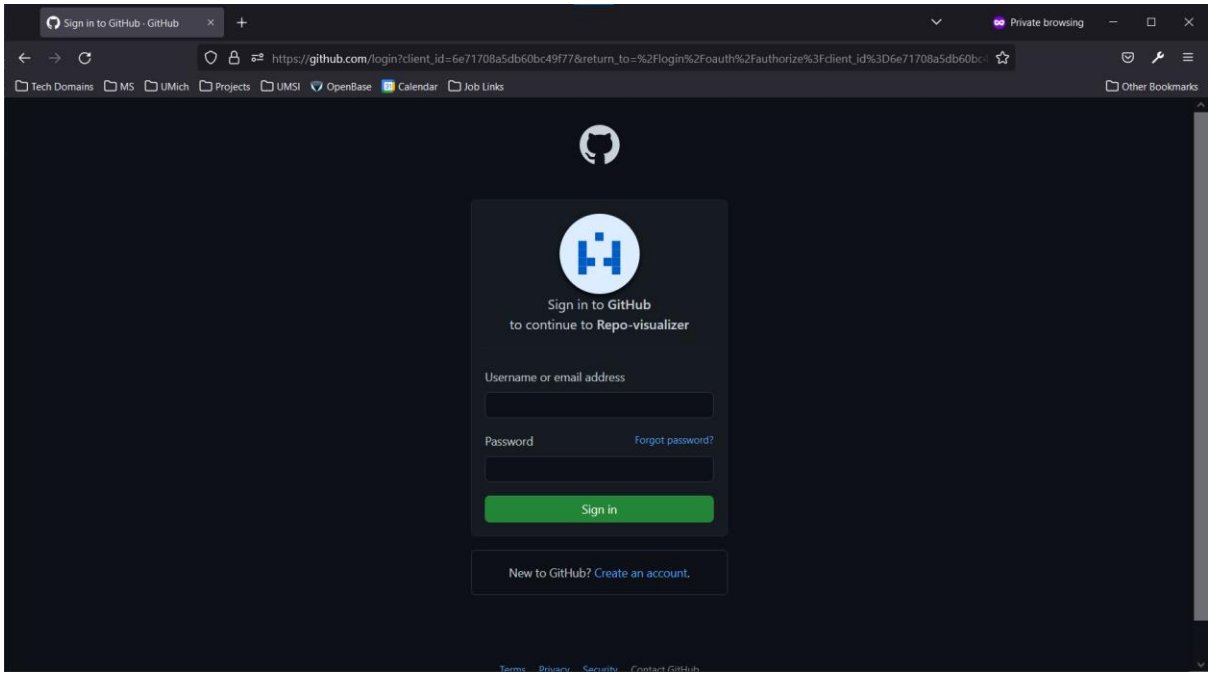
Plan of organizing data into structures

I plan to fetch repository commit histories and depict a repository visually in the form of a tree. I intend to first implement this for individual branches, before attempting to show how branches flow out of the main/master branch through a comprehensive visualization.

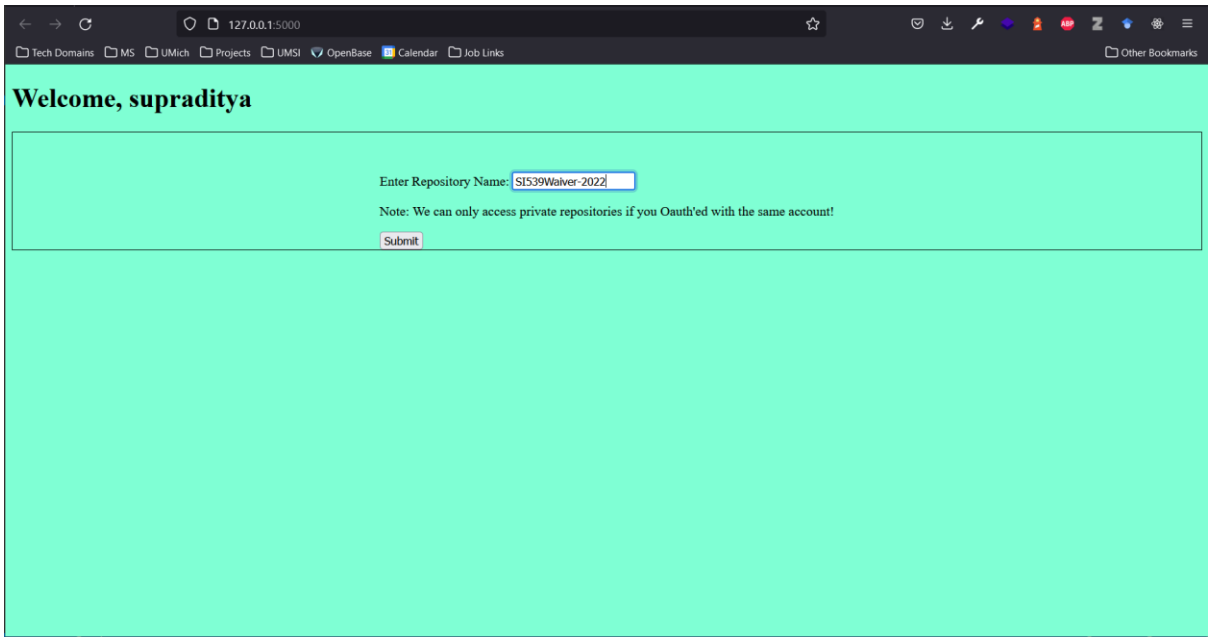
The user will be able to select any repository from within their account, choose between being able to view a visualization of a single branch versus the entire repository, and then hover over individual commit nodes to get an overview of what said commit is about.

I plan to use flask along with some CSS to implement my visualizations. The end-product should look like a top-to-bottom tree.

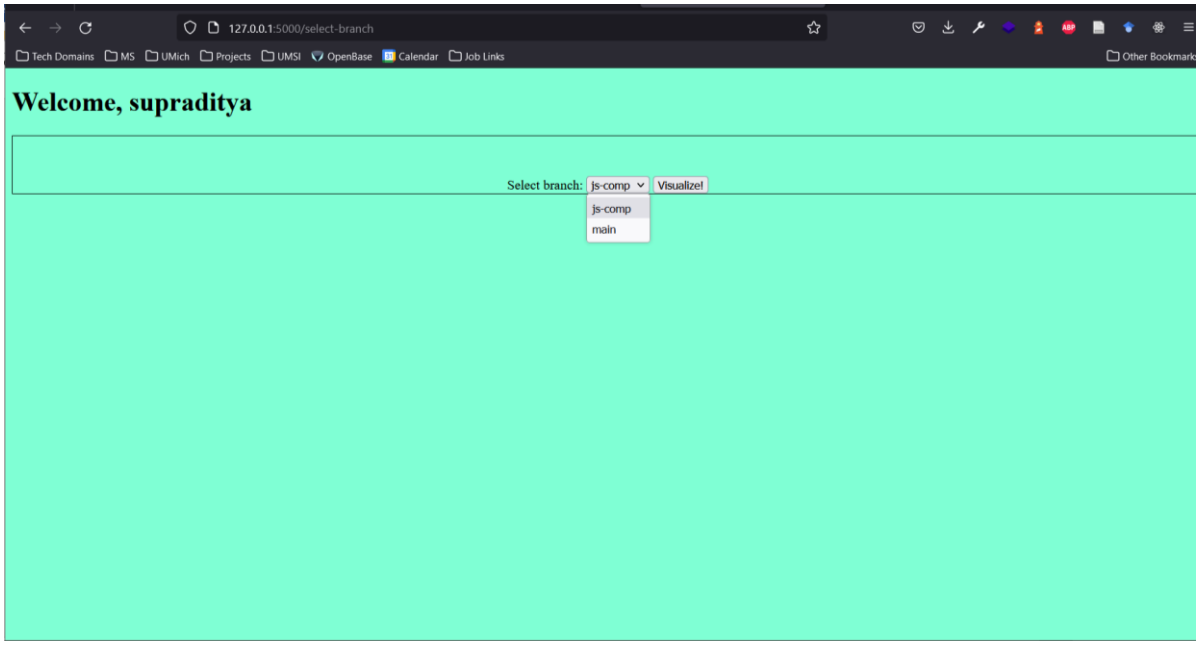
Some in-progress screenshots



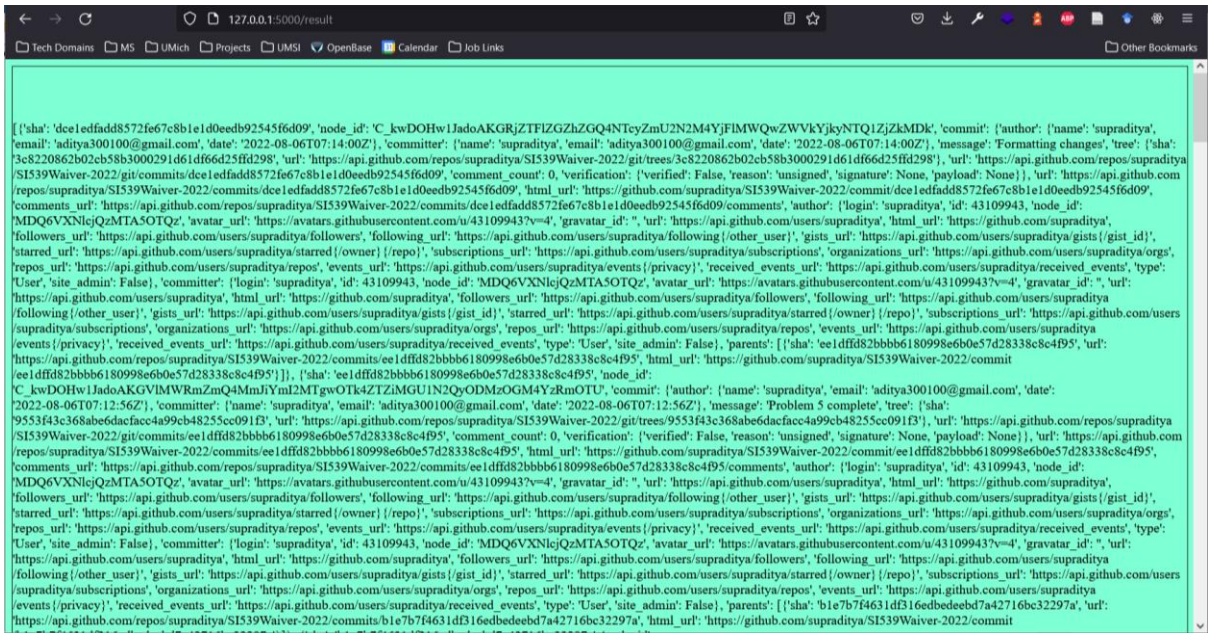
OAuth implementation



Post-OAuth greeting (the username is fetched via API!)



Dynamically fetching all the branches within the entered repository name



Fetching of relevant API data (work-in-progress)