SI 507

FINAL PROJECT PROPOSAL

NAME: ADITYA SUNIL MENON

SELECTED WEB API: GitHub API (https://docs.github.com/en/rest) (Requires OAuth)

PRELIMINARY DATA SOURCES: Specific repositories on Github (Single or multiple pages to be scraped). Intended for user to enter which repository they wish to work with.

OUTLINE OF IDEA: *Visualizing the structure of a repository's commit history, branches, languages, and technologies used. Intended to give the user a bird's-eye view of how the repository came together over time.*

In large-scale projects and software development endeavours, it is common to see multiple branches being created, merged, rebased, and refactored over time. Several commits might overwrite certain pieces of code, cause merge conflicts, and amidst all this chaos, in comes the unassuming intern, ready to try his hand at the task assigned to him. The problem here is that while they might know how Git and GitHub work, it is very difficult to visualize branches which themselves originate from other branches, instead of being directly attached to the master branch.

The only tool available to this intern is his knowledge of the command line. By using commands such as "git log –oneline" and "git branch -a," they can attempt to understand where their code lies in the grand scheme of things, and how they are supposed to proceed with respect to commits and the eventual merge. However, the lack of preparation for the real-world often leads to these same interns committing grave mistakes while pushing their code, often causing dreadful merge conflicts, or deep-rooted errors from which a large-scale project could take days (if not weeks) to recover from.

Thus, in order to bridge this gap between the intern's knowledge of small-scale Git usage, and its implementation in large-scale multi-user collaborative initiatives, I aim to develop a visual way of traversing through a repository's commit history, where a user can see how the branches physically relate to each other. In this unidirectional directed graph network, individual commits will be nodes, and linked nodes will depict branches. A novice user will be able to use my application to perform operations that they otherwise will have to use terminal commands for (creation of branches, rolling back to certain commits) by simply pointing and clicking on commit nodes, and implementing version control in a unique and intuitive way.

Example: