Gitlytics: A Framework for Analyzing Git-based Software Development Collaboration

Jake Gutierrez
Department of Math and Computer Science
Southwestern University
Georgetown, Texas, USA
gutierr8@southwestern.edu

Dr. Dilma Da Silva
Department of Computer Science and Engineering
Texas A&M University
College Station, Texas, USA
dilma@cse.tamu.edu

***Abstract*—Github is a site where users can host code repositories using the Git version control system. Github is widely adopted in industry and in project-based undergraduate courses such as Software Engineering or Senior Design Capstone. Github provides analytics for public repositories for its main site and Enterprise editions. When instructors require students to utilize Github for assignments, they can leverage data analytics to reveal patterns in how students develop their solutions. They can also detect patterns of team collaboration requiring intervention. However, making repositories public is not desirable in academic settings due to plagiarism. In this project, we develop a framework to advance the use of Github in academic settings. We developed a web application to visualize multiple Github repositories in one place.**

I. Introduction

Github is a popular code hosting website utilizing the Git version control system. Github provides user profiles to see each other's code repositories as well as an issue tracker, branch management, forks and merging one repository to another via pull requests. A team utilizing Github would benefit from having their code to easily see how much each team member is contributing to their repository. Github also provides an interface where members can see how much each of their members is contributing in the form of additions and deletions of lines of code.

Github's usefulness has proved itself in higher education by being used in high-level, or even lower-level [1], undergraduate courses. However, when faculty use Github they find themselves wondering how can they see contribution without having to go to every individual repository. Additionally, private repositories might be needed and Github does not provide analysis and visualization if the repository is private. As a solution, a web application was created to collect Github repository data to easily visualize collaboration.The data is collected into a repository that can integrate other data sources related to student code development activities.

II. Previous Work

Github recognized that universities were using its services in their courses so they created Github Classrooms. Classrooms provides functionality similar to that of a learning management system, but with support to track and generate repositories specific to a classroom with support for individual and group repositories. Classrooms also has auto-grade functionality. This allows faculty and TAs access to private repositories to see how students are working, however it does not provide analytics like a public repository. Additionally, Github Classrooms only works with Github's main site and not Github Enterprise, i.e., instances of GitHub hosted within the university's network.

Regarding previous work in academia, there has been work done by analyzing GitLab (a service similar to Github) repositories[2]. The purpose was to analyze GitHub data to find gaps in the student's software engineering skills. The researchers processed data from a Continuous Integration/Delivery framework such as Jenkins, issue tracker data from GitLab and data from the Git repository itself.

There was previous work done in this area at Texas A&M University[3]. A script was used to pull data from specific Github Enterprise repositories. The problem with this method was that the repositories must have followed a specific naming scheme so that the script can find it. These naming schemes were not followed some of the time resulting in incomplete data. This work is the primary inspiration for this research.

There is also another web application under the same name "gitlytics" where the main focus is on industry users of Github or another service such as GitLab or Bitbucket. This website was able to pull Git repositories from those service or from a custom URL and run analytics and visualize them. A potential problem with this method is that it has to download a whole repository, which may be large (>500 mb) in some cases and would be costly in a production environment. However, the idea of using any Git repository is something to look into in the future. Additionally, the specific use case of this project is to use in conjunction with Texas A&M University's Github Enterprise server.

III. Design and Implementation

The "gitlytics" for this research is a full-stack web application. This means that there are at least two different parts, a front-end and a back-end. The front-end is responsible for taking information from the back-end and displaying it to the end-user in a friendly manner. The back-end takes data from the database and turns it into a response that the front-end can understand. While there are front-end libraries that can communicate with a database, this project required functionality from a back-end server that the front-end can not provide.

A. Front-end

The front-end of this project was created using React.js[1], a popular JavaScript framework created by Facebook. React.js offers several data-visualization libraries for the framework. To speed the development process, we used Material-UI[2], a UI framework for React.js based on Google's Material Design specification.

B. Back-end

The back-end communicated with the front-end via a REST (REpresentational State Transfer) API. This means that it uses JSON documents to send information to the front-end. REST is a standard used by a majority of companies and is what Github uses for their API. The back-end is written in Python using a micro-framework called Flask[3]. Flask is small and provides plenty of extensions via PIP modules to build a prototype for this project.

For the database, going with a relational database would make the most sense due to the relational nature of our database design, from the courses and projects to the students and groups. For a database, PostgreSQL[4] was used since it provides aggregation functions and other functions helpful for our purposes.

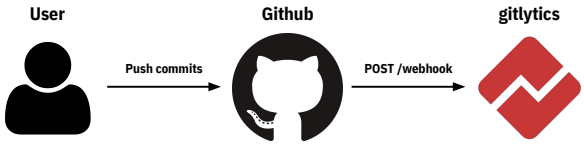

Fig. 1. Example of a webhook when a User pushes to a repository

To get data from Github, we use the user's Github credentials to get their repository commit data. Github webhooks are also utilized to get commit data whenever a user pushes commits to a repository. Using webhooks avoids having to actively call Github's API periodically for updates (e.g., every hour or so.) Calling Github's API for commit data has proved to be time consuming as we have to call the API for each individual commit for each repository to get the necessary data. Since getting commit data is time consuming, the server uses Celery[5] along with Redis[6] to run background tasks as to not lock up the main server.

C. Design

The database design closely resembles a typical learning management system design. Users can create courses and invite existing Github users to them. In a course, they can create projects with a name, description, start and end date, and a type. Types tell us how the students will be working in their repositories whether it is individual or group. If a group type is specified, then they can determine groups through a drag-and-drop interface. Once a course is created, students can assign their Github repository as a group or individually, depending on the project type. When a repository is assigned, a Github webhook is created for the repository.

Whenever a webhook is sent to the back-end, commit's SHA (a unique ID) and repository name is stored. After the initial commit data is stored in the database, a background task is called through Celery to get more information about the commit such as the additions and deletions.

[1]https://reactjs.org
[2]https://material-ui.com
[3]https://flask.palletsprojects.com
[4]https://www.postgresql.org
[5]https://docs.celeryproject.org
[6]https://redis.io

D. Implementation

The front-end uses a data visualization library called Recharts[7] which provides functionality for making very simple to complex charts. The library also supports legends and synced tooltips. Synced tooltips are a way to see multiple tooltips on multiple charts while hovering a cursor on just one. For instance, there is a chart to visualize the number of commits for each contributor per day, and there is a chart for the number of additions and deletions for a contributor per day. The x-axis for both have the same date-range and the same number of points. While one user is hovering over the chart to see the number of contributions, they can simultaneously see the number of additions and deletions a contributor has without having to go back and forth between the two. An example of the interface is shown in Fig. 2.

The back-end utilizes pandas[8], a popular analytics library for Python. Pandas was used to process data from the database in a format that Recharts can understand.

IV. Future Work

As of the moment, "gitlytics" is running on an AWS Lightsail virtual private server. AWS Lightsail provides cheap VPSs that are great for small to medium sized projects. "gitlytics" is running using Docker Compose, a helpful development tool utilizing Docker containers. Docker Compose is not recommended for production use since it can not run more instances of the same container. In order to scale well, an option would be to use AWS Elastic Beanstalk, a service similar to Docker Compose but it provides automatic scaling of its' containers. Another solution would be to use Kubernetes[9], a popular container orchestration software. Kuberenetes utilizes Docker containers into "pods" that can be scaled automatically. Kuberentes has proved itself in production environments[4] so it would be ideal to eventually move to Kuberenetes.

One of the reasons Flask was used is that it provided easy to use OAuth2 libraries. However if these libraries were used, we could not use a front-end Javascript framework like React.js since we would be using Flask's HTML rendering engine. So we started using Flask expecting to use the rendering engine, however it proved to be difficult as it was not as flexible as React.js. So Flask was used despite not using the rendering engine anymore. If we were creating a REST API from the start, another Python web framework would be used called FastAPI[10]. FastAPI is specifically for creating REST APIs and is faster compared to Flask.

V. Conclusion

This report details the work of a web application that collects collaboration data from Github to be viewed by faculty to look at collaboration of a programming assignment, similar to that of a Capstone project. The web application is full-stack with React.js as the front-end and with Flask as the back-end. To collect data, GitHub's API and webhooks were utilized to get commit data for each repository. The data is then process and is able to served to the front-end via a REST API and visualize using Recharts. The web application supports logins via Github, course and project creation.

Acknowledgment

The work of Jake Gutierrez was supported by the Distributed Research Experiences for Undergraduates (DREU) program, a project of CRA-W and the Coalition to Diversify Computing (CDC). Jake Gutierrez would like to thank AccessComputing, a sponsor for the DREU program.

References

[1] G. Sprint and J. Conci, "Mining github classroom commit behavior in elective and introductory computer science courses," *J. Comput. Sci. Coll.*, vol. 35, no. 1, p. 76–84, Oct. 2019.

[2] J. C. C. Ríos, K. Kopec-Harding, S. Eraslan, C. Page, R. Haines, C. Jay, and S. M. Embury, "A methodology for using gitlab for software engineering learning analytics," in *Proceedings of the 12th International Workshop on Cooperative and Human Aspects of Software Engineering*, ser. CHASE '19. IEEE Press, 2019, p. 3–6. [Online]. Available: https://doi.org/10.1109/CHASE.2019.00009

[3] C. M. Smith, "A toolset for mining github repositories in educational software projects," Aug 2018. [Online]. Available: http://hdl.handle.net/1969.1/173656

[4] J. Shah and D. Dubaria, "Building modern clouds: Using docker, kubernetes google cloud platform," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, 2019, pp. 0184–0189.

[7] http://recharts.org

[8] https://pandas.pydata.org

[9] https://kubernetes.io

[10] https://fastapi.tiangolo.com

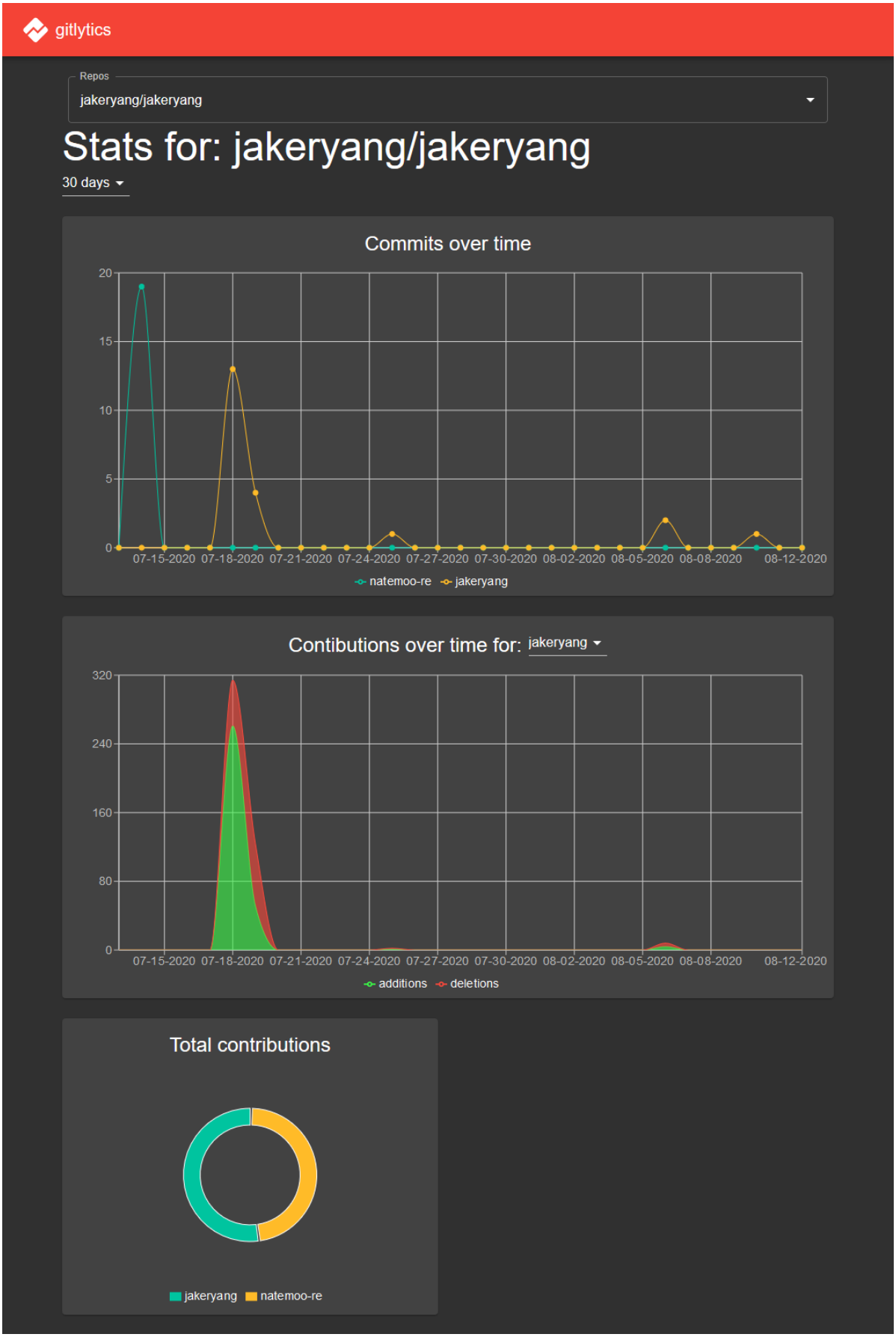

Fig. 2. A screenshot of the "gitlytics" interface. A dropdown at the top allows faculty/course owners to switch between multiple repositories within a project. Students will be able to see this page without the dropdown and will see their assigned repository be default. The first chart provides a simple number of commits over time per contributor. The second chart is an area chart for the number of deletions and additions (per day) over time for each contributor. One can change which contributor they are looking at by the dropdown next to the title. The third chart is a donut chart to see commits for each contributor.