Github Wrapped Project Proposal

***Abstract*— A project proposal for a program for developers to reflect on their annual Git history, providing insights into their development patterns and design choices in an interesting way.**

I. Project Objective

GitHub Wrapped is a program for developers to reflect on their Git history, providing insights into their development patterns and design choices in an interesting and unique way.

The application's main purpose is to provide an entertaining way for users to experience their GitHub contributions and activity over the past year. Additionally, this will boost relationships in the open-source community by providing an expressive outlet for users to show their development activity. Furthermore, GitHub Wrapped can boost engagement with GitHub services and increase willingness to purchase premium plans and products to gain access to these insights. This has been shown to be true in the case of Spotify Premium where Spotify Wrapped has been something that many users are willing to purchase for the community experience.

II. Motivation

A. Provide users with a comprehensive overview of their GitHub activity over the past year

Often, developers will have hundreds or thousands of contributions in a year. For this reason, it is quite difficult to gain a broad overview of one's activity with the massive amount of data that accumulates over the year for contributions. Furthermore, these contributions can range across many repositories with vastly different purposes and objectives. Unfortunately, GitHub does not currently provide a method of viewing this information at a macroscale; rather, it requires combing through days of contributions on your profile. GitHub Wrapped provides a solution by consolidating this information into one place and providing a fun way to experience this review of a developer's year.

B. Facilitate better understanding of personal development progress and contributions

GitHub Wrapped will provide an entertaining way for users to explore their achievements and progress throughout the year. This will include an analysis of the new domains they have explored, recent technologies they used, and causes for which they have shown passion. These meaningful insights will be valuable for active developers who would like to reflect on their experiences in the past year. Moreover, GitHub Wrapped offers a unique opportunity for users to strengthen team dynamics and foster closer connections within friend groups, and the broader open-source community.

III. Data Sources and Scope of Work

A. User authentication and authorization

GitHub Wrapped will provide GitHub OAuth integrations so users can authenticate themselves against their personal GitHub account. Additionally, this integration will request the necessary permissions from the user so that our application can retrieve information from the GitHub API. It will be important to limit these permissions as much as possible to avoid limiting the number of users who are uncomfortable with our third-party service.

B. Data retrieval from GitHub API

After GitHub OAuth has been integrated, our application will be capable of viewing the complete contribution history of the current user. The GitHub REST API offers extensive access to specific endpoints, granting access to a wide range of information, including details about commits, collaborators, repositories, and many other resources. GitHub's documentation provides all the available endpoints.

C. Data visualization and analysis

Analytics and visualization form the foundation of GitHub Wrapped. The project aims will capture valuable and creative metrics such as preferred programming languages, topics of interest, and the closest collaborators. Additionally, GitHub Wrapped will feature playful metrics, such as determining one's "developer spirit animal." The analysis to derive these playful metrics will be slightly more arbitrary; however, it offers the challenge of producing something meaningful with a plethora of concrete information sources provided by the GitHub API. To package these analytics, we intend to create a vibrant and engaging interface for users to explore their GitHub Wrapped experience.

IV. Stretch Goals

A. Third-party integrations beyond GitHub API

As a stretch goal, we plan on integrating additional third-party services or APIs beyond the GitHub API. These integrations could provide users with extended capabilities or access to data from other platforms, potentially enriching their overall experience.

B. Advanced analytics or machine learning components

As another potential stretch goal, we plan on researching the feasibility of implementing advanced analytics or machine learning components within the application's initial release. These components could offer users more sophisticated data insights, predictions, or personalized recommendations based on their GitHub activity. This goal aims to elevate the application's value proposition by leveraging new technologies.

V. Member Responsibilities

- Logan O'Neal - Language Popularity Analysis
- Eric Vaughan - Commit Analysis
- Jacob King - Mapping User Network
- Hayden Curl - License Popularity Analysis
- Caleb Fisher - Developer Spirit Animal

VI. Milestones

A. Milestone 1: User Authentication and GitHub API Integration (Weeks 1-2)

During this initial phase, the focus will be on setting up user authentication, allowing users to securely log into the web application. Additionally, integration with the GitHub API will be established, enabling the application to access and retrieve user-specific data from their GitHub account.

B. Milestone 2. Data Retrieval and Processing (Weeks 3-4)

Next, the team will concentrate on retrieving relevant data from the user's GitHub account. This will involve implementing mechanisms to fetch and organize information such as commit history, repository activity, and among other insightful metrics. During this phase the team will also focus on cleaning and processing procedures in order to ensure accuracy and consistency in the final application.

C. Milestone 3: Activity Overview and Initial Data Visualization (Weeks 5-6)

During weeks 5 and 6, the team will focus on building the UI for visualizing the git data. Initial data visualization techniques will be applied to present key metrics and trends, providing the user with a clear and comprehensible snapshot of their GitHub activity over the past year. The exact details of the visualizations should be clearer after completing the previous milestone as the team will have a better understanding of the available data.

D. Milestone 4: Finalizing User Interface and Testing (Weeks 7-8)

The final phase will involve refining the user interface to create an intuitive and user-friendly experience. Feedback from user testing will be incorporated to fine-tune the application's design.

VII. Results

A. Developer Spirit Animal

After utilizing a variety of metrics such as repository names, favorite languages, and recent contribution activity, a GPT-4 prompt was able to provide insightful reasoning for the selection of spirit animals. Additionally, we were able to get sufficient levels of randomness by engineering the prompt in such a way that it promoted creativity. Unique spirit animals such as the "chameleon", "platypus", "salamander", "lynx", and "armadillo" were among the most popular for the developers that were tested.

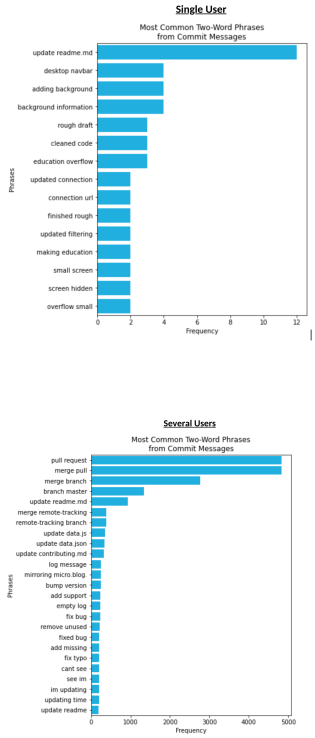
Also, the spirit animal visualization was able to further capture the personality of the developer by supplying the DALL-E 3 prompt with rich information from the profile README and bio. Often, DALL-E 3 recognized the importance of school and career information in forming the image of the developer's spirit animal. Interestingly, DALL-E 3 often assumed that developers should have items such as hoodies, coffee mugs, and headphones. One of the generated images for the platypus spirit animal can be seen above.

B. Commit Message Analysis

The purpose of analyzing GitHub commit messages was to determine a user's most common contributions on GitHub. To provide a point of comparison, commit messages from several users across GitHub were also examined. The message analysis involved three main steps: collecting, preprocessing, and counting the data. First, for the single user's analysis, the commit messages were collected simply using the GitHub REST API, while for the several users' analysis, Google's BigQuery was used to quickly obtain a large number of messages. After being collected, the messages had to be preprocessed, which involved dividing the messages into consecutive two-word phrases. For example, if a commit message was "sample commit message here," it would be broken down into "sample commit," "commit message," and "message here." Lastly, the frequency of each two-word phrase was counted.

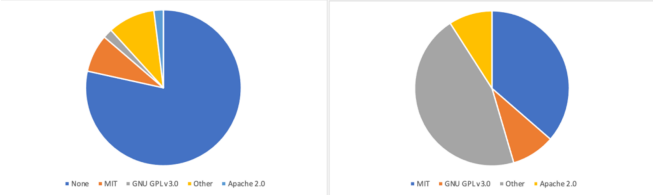
The two figures below depict the most common GitHub contributions from the perspectives of both an individual user and a multitude of users. For the individual user, the biggest conclusion that can be drawn from the graph is that they are likely very meticulous about keeping their README files up to date. In terms of users from across GitHub, there

are five contributions that stand out as being by far the most common, and they are not particularly surprising. The first four pertain to pulling and merging, and the fifth most common contribution, similarly to the individual user's top activity, refers to updating README files. Some of the other common actions listed in the graph for several users include updating data, fixing bugs, removing unused parts of code, and adding missing details in code.

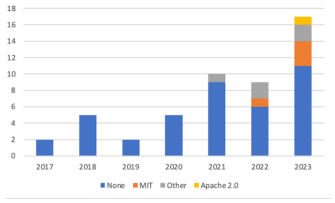


C. How have the most common licenses changed over time?

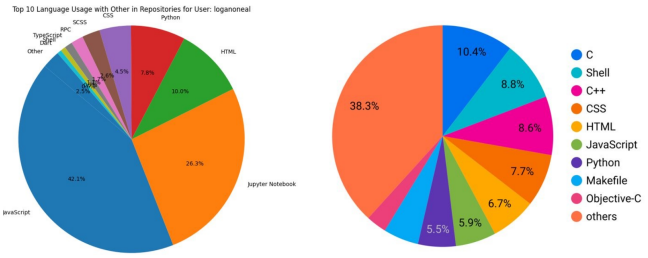
To find how the most common licenses changed over time, we performed an analysis of all our group members' public repositories. To accomplish this, we utilized GitHub's API functionalities to retrieve all of the repositories' data in a json response. After parsing through the data, we were able to access information about the license, if any, such as the license key, the license name, and the URL that leads to the license file. For our purposes, the type of license was determined by the license name. Our data shows that for our repositories, the overwhelming majority did not have a license. For the repositories that did, most were either MIT Licenses or licenses listed by the GitHub API calls as "Other". Below is a graph that shows how our repositories



changed over time. For the first several years, none of our repositories were licensed. However, as we have progressed through college, we can see that more and more of our repositories are licensed. This is largely due to having been assigned more complex projects or labs and tackling more challenging personal projects. Additionally, we are utilizing more free and opensource GitHub repositories to assist us in our work.



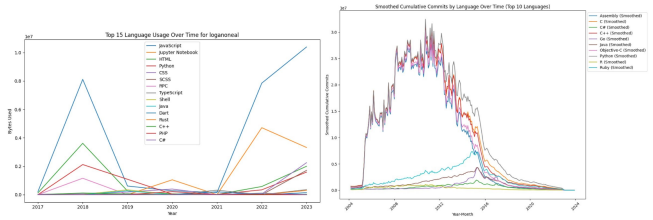
D. Github Language Popularity



For a single user, the analysis involves querying their repositories via the GitHub API to collect information on the languages used. This data can then be processed and aggregated to identify the user's most frequently utilized languages. For instance, if a user predominantly employs JavaScript, Python, HTML, and CSS across their repositories, this insight can be extracted through the API's repository data.

On a broader scale, examining GitHub as a whole entails querying a vast repository of code using BigQuery, allowing for a comprehensive analysis of language usage trends across the platform. Identifying the most popular languages involves aggregating data from numerous repositories to determine the prevalence of languages like C, Shell, C++, and HTML. BigQuery's capabilities enable the extraction of these language preferences by analyzing the frequency and distribution of languages in repositories across GitHub.

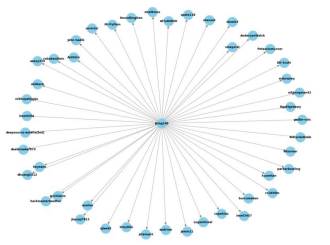
By combining the GitHub API's repository-specific data retrieval with BigQuery's powerful data analysis capabilities, we can gain a nuanced understanding of language popularity both at an individual user level and across the expansive GitHub ecosystem. This analysis provides valuable insights into programming language preferences, aiding developers and organizations in making informed decisions regarding language adoption, resource allocation, and community engagement. Analyzing the language trends of a single user over time versus the broader GitHub landscape gives insight into evolving programming preferences. When examining an individual user's language usage, the GitHub API facilitates tracking changes in their coding repertoire across



various repositories, with each language being scaled by the number of code bytes written in that language.In contrast, language trends across the entirety of GitHub showcase broader industry-wide patterns and preferences. However, there's an intriguing observation when comparing the trends: an apparent downward trajectory in language use across the platform over time. This downward trend can be attributed to a limitation in available BigQuery data for more recent years. The decrease in apparent language usage isn't indicative of a decline in programming interest but rather a lack of comprehensive data representation in the recent years' dataset within BigQuery. Therefore, the apparent decline in language use across GitHub in the data doesn't align with the actual programming landscape and shows the necessity for updated or expanded datasets within BigQuery to accurately reflect the current trends in language usage.

E. Mapping User Github Network

The purpose of obtaining a network map for Github users is to help users see all of the people that may have collaborated with or crossed paths with on Github. As developers we work with so many people on every project that we often don't even realize. Because of this we wanted Github Wrapped to be able to not only show a user all of the people they have worked with but also them by using a node tree.



To obtain the data we first needed to find all the repos that a user contributed to. To do this we first used the Github API to query for all public repos that a user has forked or contributed to. After we had obtained all repos that the user had contributed to them we could look through all of those repos and find other people who have also contributed to those repos. This in turn gave us a list of users that the user of Github Wrapped had crossed paths with. Then by using matploit and python we could create a node tree that looked like the figure above.

F. Primary Issues Encountered

During the development of the GitHub Wrapped project, the team encountered several challenges, one of which involved difficulties in implementing GitHub OAuth for user authentication. This issue required close collaboration with GitHub API documentation to address potential token problems and ensure a secure and smooth user authentication process. Additionally, API rate limiting posed another challenge, particularly for users with extensive contribution histories. To resolve this, the team implemented caching mechanisms, optimized data retrieval queries, and handled rate limit errors gracefully.

Another issue arose from inconsistencies in the GitHub API data, impacting the accuracy of visualizations due to data cleaning challenges. The team tackled this problem by developing robust data preprocessing algorithms, handling edge cases, and implementing data validation checks to ensure data integrity. The complexity of representing diverse GitHub contributions in a visually appealing and understandable way also presented challenges. The team addressed this by iteratively designing and testing visualizations, simplifying complex metrics, and gathering user feedback to enhance the overall user experience.

VIII. Future Work

A. Enhanced Machine Learning Integration

Exploring advanced machine learning components remains a potential area for future development. These components could offer users more sophisticated data insights, predictive analytics, or personalized recommendations based on their GitHub activity. By leveraging machine learning algorithms, GitHub Wrapped could provide users with even more valuable and tailored information about their coding patterns, preferences, and potential areas for skill development.

B. Extended Third-Party Integrations

As a stretch goal in the initial proposal, further exploration of third-party integrations beyond the GitHub API could enhance the overall user experience. Integrating with additional platforms or services could provide users with extended capabilities, diversified insights, and a more comprehensive overview of their development ecosystem.

IX. Conclusion

In conclusion, the GitHub Wrapped project provides developers with a unique and engaging platform to reflect on their annual Git history. Throughout the development process, the team encountered and successfully navigated various challenges, from implementing secure user authentication and overcoming API rate limits to ensuring data integrity and crafting meaningful visualizations.

To conclude, our project aims to revolutionize the way users engage with their coding journey. With its intuitive interface and seamless navigation, the platform offers more than just a progress report. It serves as a powerful tool for self-improvement, providing tailored insights into areas where users can further enhance their software development

skills. GitHub Wrapped ensures a smooth and responsive experience across various devices, allowing users to effortlessly access their personalized data and track their growth. This accessibility empowers developers to stay connected to their coding endeavors no matter where they are, fostering a sense of continuity and progress in their software development journey.