PMPro - Cal Poly Senior Project

CARMINA CRUZ, Department of Computer Science Software Engineering
LOUISE IBUNA, Department of Computer Science Software Engineering
DR. BRUNO DASILVA, Department of Computer Science Software Engineering

ACM Reference Format:
Carmina Cruz, Louise Ibuna, and Dr. Bruno daSilva. 2020. PMPro - Cal Poly Senior Project. 1, 1 (June 2020), 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In software engineering, platforms such as GitHub are used by developers to store projects and network with like-minded people. From small class projects to large projects that reach millions of users, GitHub is a notably used platform to publicize projects by using repositories, as well as version control to collaborate with other users. One of the features GitHub provides is pull requests. Pull requests are basically a gateway to creating a revision on the original repository and reviewers can see if they would like to accept or decline the changes to the master branch. This is an essential feature, especially in software teams who are constantly implementing code for an application and need to add it to the project. Reviewers can see if the code looks good depending on requirements and can either accept or request changes to the pull request. Other times, they can even reject a pull request. That's what we wanted to achieve in this project.

On this work, we propose PMPro – a tool to analyze pull requests by simply inputting a GitHub repository and analyzing data through graph visualizations. The ultimate goal we want this tool is to provide data analysis features by pulling data from GitHub repositories and visualizing them on a dashboard. Companies rely on this to do informal code reviews and ensure that their developers are writing clean, quality code. By utilizing this tool, we can see a project's trend on pull requests and determine what these numbers associate to. Is the repository owned by a company that has requirements set in terms of code quality? Would this be due to the size of each pull request that needs to get reviewed? These are some of the many questions we want to solve for software teams so that they can bring the analysis back to their team.

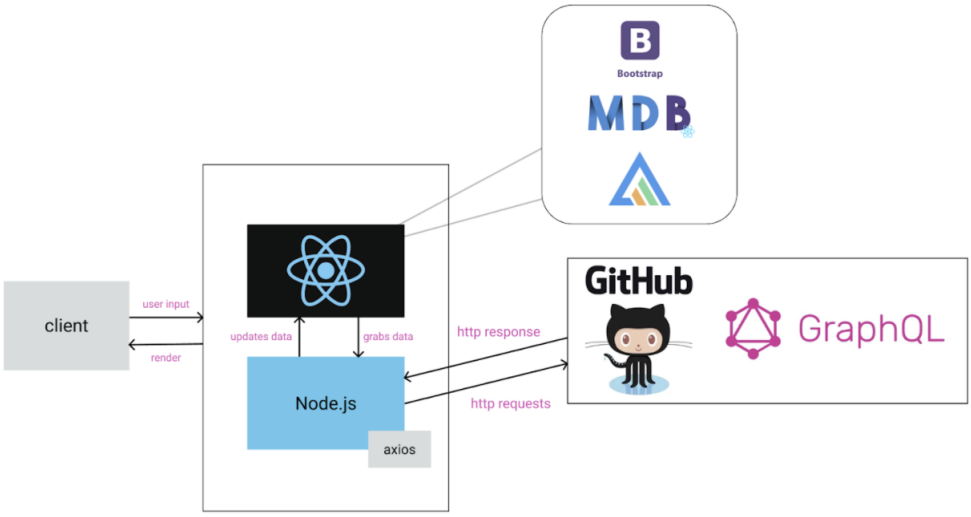
2 OUR SOLUTION

Our solution's target outcome is to be able to provide a pull request dashboard that would allow software engineers to be able to track their team's health based on pull request content, such as code and conversations via comments. In order to create the dashboard, we decided to develop a web application using a React-Node.js-GraphQL tech stack. The backend of the app uses Node.js and Axios to send HTTP requests to the GitHub GraphQL API server

Authors' addresses: Carmina Cruz, Department of Computer Science Software Engineering, 1 Grand Ave, San Luis Obispo, CA, 93405, ccruz27@calpoly.edu; Louise Ibuna, Department of Computer Science Software Engineering, 1 Grand Ave, San Luis Obispo, CA, 93405, libuna@calpoly.edu; Dr. Bruno daSilva, Department of Computer Science Software Engineering, 1 Grand Ave, San Luis Obispo, CA, 93405, bcdasilv@calpoly.edu.

XXXX-XXXX/2020/6-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

in the form of a GraphQL query, which then responds with the query-specified data. This data is then stored in the graph component's state, which is then rendered on the front end using React. We use Bootstrap, MDB React, and ApexCharts.js for the front end to render the data into the pie and bar charts.



3 CONCLUSION

Our project's goal was to analyze pull request trend's in GitHub repositories using an application. We can do this by inputting information about the repository and outputting data through a graph visualization. For the past two quarters, we were able to create a dashboard for the user to look at the data and search for a repository using a search bar. We also connected the front end and the back end so the application can search for the repositories. This is triggered by utilizing HTTPs requests with GraphQL and would return results the user wants. This is a good start to the overall goal of our application. In the future, there are some features that would like to be added to the application. Some of these features include checking for the user who reviewed the pull request, using different types of graphs, and check pull requests with request changes. We hope that once these features are added, it would be in production and used by many software teams to create high quality pull requests.