RepoRanger: Streamlining Collaboration for Software

Engineering Teams

The Bit Co.

Kanad Naleshwarkar

Virginia Tech   
 kanadn@vt.edu

Sanket Bhujbal

Virginia Tech   
 sanketb@vt.edu

Sarthak Banerjee

Virginia Tech   
 sarthakb@vt.edu

ABSTRACT

*In today's world, software development teams are often spread out across different locations, making remote collaboration tools more important than ever. One popular tool for team communication is Discord. However, managing a project's repository and issues often requires switching back and forth between Discord and GitHub, which can be time-consuming and inefficient. To address this issue, we propose the development of RepoRanger, a bot that leverages the Discord and GitHub APIs to allow team members to manage the project repository, its collaborators and the project board all from a Discord server. RepoRanger will enable team members to add new collaborators, review pull requests, create and list issues, and initiate workflows, as well as monitor project builds and deployments. With RepoRanger, software development teams can streamline their workflow and increase efficiency by reducing the time and effort needed to manage repositories and track issues. This project represents an innovative solution to the challenge of remote collaboration, providing teams with a more seamless and integrated communication channel.*

1. **INTRODUCTION**

Remote work induced by the pandemic has increased software engineering teams’ reliance on communication apps like Slack [1], Discord [2] and Teams [3]. Even though these apps provide an excellent collaborative platform, there are many aspects of project management that can be improved. For example, it is particularly difficult for DevOps managers in a team to quickly get info on how long it takes for a new release of the product to be built and deployed on servers, especially if the manager lacks technical knowledge. Similarly, with the lack of a physical Kanban board, project managers find it troublesome to go back and forth with issue-tracking portals while discussing open issues with the team.

We believe that such problems can be solved, not by developing new software, but by integrating existing tools so that the combination can provide more value to engineering teams. Therefore, we propose an intuitive integration of GitHub [4] and Discord, and we call it ‘RepoRanger’. This application will provide an interface for team members, by means of a Discord bot [5], to access their GitHub repositories, projects as well as CI/CD pipelines (workflows) without having to switch between various portals.

This integration, for instance, can help managers get notified about recent issues, pull requests, etc., or fetch a particular issue from the backlog so that it can be discussed with the team in the same Discord channel. The RepoRanger bot will facilitate this communication by providing a set of slash commands [6] that abstract complex API calls and git commands, making it easier to quickly get info and perform operations on the user’s behalf. Hence, we believe RepoRanger will improve efficiency, reduce errors, and ensure that everyone is on the same page.

1. **BACKGROUND**

In the world of software development, version control and project management play a crucial role. GitHub, a popular web-based hosting service for version control and collaboration, has become a standard in the industry. It offers various features such as bug tracking, task management, and wikis, making it a comprehensive platform for developers to work together.

Discord is a communication platform that provides voice, video, and text chat services. It has become popular among gaming communities and software development teams due to its ease of use and flexibility. Discord also offers an API that allows developers to integrate its chat services into their own applications.

API stands for Application Programming Interface. It is a set of protocols, routines, and tools that allow developers to interact with a software application or service. APIs enable applications to communicate with each other and share data seamlessly.

Replit is an online integrated development environment (IDE) that provides a platform for developers to write, test, and deploy code directly from a web browser. It offers various features such as version control, collaboration, and hosting, making it a popular choice among developers.

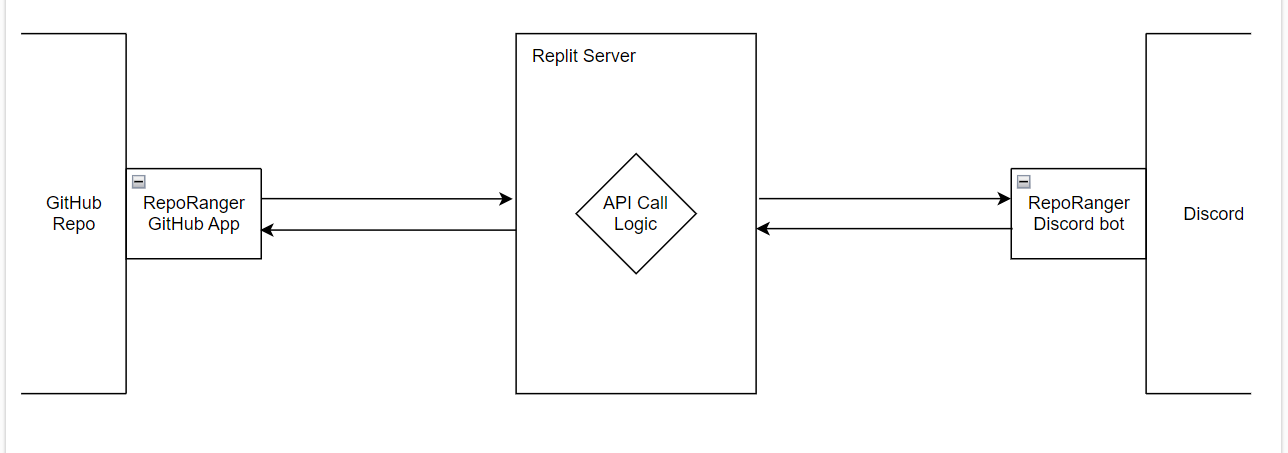
The integration of these technologies can provide developers with a powerful toolset to manage their projects efficiently. By combining GitHub and Discord, developers can communicate effectively and collaborate seamlessly. The use of APIs can further enhance integration by allowing developers to automate various tasks and streamline their workflows. Hosting the integration on Replit provides a convenient and accessible platform for developers to work on their projects from anywhere.

1. **RELATED WORK**

A similar integration for Slack and GitHub has already been developed [7]. Also, there are many services like Pipedream [8] that provide integration between GitHub API and the Discord Webhook API. Inspired by these solutions, we developed RepoRanger with the aim of making it easily configurable with any GitHub repository and project. Our goal was to create a user-friendly, efficient bot that could be customized to suit any team's needs. By leveraging the existing work and adding our own unique features, we were able to create a tool that streamlines the communication and collaboration process for GitHub-based projects.

Figure 1: Architecture diagram for RepoRanger

1. **IMPLEMENTATION**

RepoRanger is based on the event-based architectural pattern. The bot uses the Discord API to listen for commands from users in a designated channel. When a command is received, the bot parses the command and uses the GitHub API to perform the necessary actions on the repository. The bot then responds to the user in the Discord channel with the result of the action.

From the design perspective, the RepoRanger project has the following 3 major components:

* 1. Discord Bot

Users of a Discord server will communicate with RepoRanger through a Discord bot which can be installed on the server. This bot can be invoked through a slash command, for example “/issuemake”. Other required inputs can be provided with this command, such as issue title and description. Then this data will be communicated, through the Discord API, with the bot implementation hosted on a Replit server where the corresponding GitHub API will be invoked.

* 1. GitHub App

GitHub Apps are the recommended way to communicate with GitHub as they offer better security. For this project, we have created a GitHub app with the same name as the bot that can be configured with any repository and with necessary permissions, this app can be used to perform operations on the repository.

For the sake of testing our implementation, we installed our GitHub app on a dummy repository called ‘RepoRanger-Playground’.

Once necessary GitHub API has been identified for the user command, there are two ways a GitHub app can invoke the API call. The first one is on behalf of a user, that is, with the help of a user’s personal access token. The second way is making the GitHub App authenticate as an installation so that it acts on its own behalf [9]. For this project, we decided to go with the first option as it provided us with more flexibility and control over the repository.

* 1. Replit Server

Finally, we needed a server to host the Discord bot and GitHub App implementation. We decided to host these implementations on Replit since it provides a fully-featured development environment that can be used directly in a web browser. Once the code is ready, it can simply be deployed and the applications start serving requests. Secondly, Replit makes it easy to collaborate with other teammates. With a shared editor, we could work on different features simultaneously.

1. **TESTING APPROACH**

Our testing approach for the Discord bot implementation involves mainly testing each feature based on the HTTP status code returned by the GET and POST requests since the implementation relies heavily on API calls. A code of 200 indicates that the operation was successful.

During the initial development phase, we ensured that the basic functionality, which includes the integration of the Discord bot with the Discord server and installation of the GitHub App on the repository, was ready for testing. After the completion of each feature, we carried out unit and regression testing in parallel with development to ensure that the latest changes are functioning as expected and that the previous functionality was not affected.

Our testing methodology ensured that we could catch errors early on in the development process, which helped to reduce the overall development time and ensured that the final product was free of critical bugs. Additionally, the testing approach also enabled us to continuously validate that each feature was working correctly, which ensured that the end product met the expected quality standards.

1. **DEPLOYMENT PLAN**

Since we are using a Replit server to host our implementation, we have two options to deploy the code. Both of them follow the basic deployment strategy wherein once code is uploaded to a Replit server, it will automatically build NodeJS components and the bot is updated instantly.

The first deployment method involves publishing our project on the Replit community page, which can be accomplished using the built-in publish function within Replit. This option would allow other developers and community members to use the bot in their Discord servers without requiring extensive installation. This approach is an excellent way to deploy our program to various developers. Moreover, Replit provides a paid feature called deployment, which permits the creator to deploy their product and keep it running 24/7 for a cost. This could be another viable option for deploying our project.

The second method involves downloading the project files as a Zip file and pushing it into GitHub. This approach would enable individuals to easily copy our program and use it wherever necessary. To use the product without any installations, a user would only need to import the Zip file directly from GitHub using Replit, providing them with the same environment.

Either of these methods can be utilized to deploy our bot, making it available for users to employ in their Discord servers.

1. **MAINTENANCE PLAN**

Our goal is to ensure that the RepoRanger bot remains functional and secure and to achieve that, we have put in place a comprehensive maintenance plan. The first step in this plan is to keep the bot up-to-date by regularly updating its dependencies. By doing so, we ensure that the bot is equipped with the latest features and security patches.

To test the bot's functionality, we will carry out regular tests to account for any changes in GitHub or Discord API policies. By doing so, we can make sure that the bot functions as intended and is not affected by any updates to the API.

We take the security of our bot seriously and have implemented measures to protect it. The authentication method used to communicate with GitHub APIs is a bearer token, which we will regenerate and replace periodically. This will prevent anyone from exploiting the token and accessing sensitive information.

1. **LIMITATIONS AND FUTURE WORK**

At present, our GitHub App relies on a personal access token of a specific user, who typically is a team lead. This implies that all repository actions are conducted on behalf of this user. Although this approach offers more flexibility and control over the repository, it also limits the bot's ability to operate independently. Therefore, we intend to modify the authentication mechanism of our GitHub App in the future to make it more flexible and not tied to a specific user.

Furthermore, our existing codebase has limited error handling mechanisms, especially for rare or unusual cases, such as when a user triggers the "/issueinfo" command, but the corresponding GitHub repository has no issues. At present, we handle this situation with a basic if statement that checks for the presence of issues in the repository. Going forward, we plan to incorporate more robust error-handling procedures that can detect edge cases and provide appropriate feedback to users.

1. **CONCLUSION**

Our experience working on the RepoRanger bot provided us with valuable opportunities to develop our skills in software engineering and project management. Through regular scrum meetings and utilizing a Kanban board to track our progress, we were able to consistently add new features to the bot. Along the way, we also gained knowledge and experience in working with API calls and learned how existing applications can create more value by exposing their functionality through API endpoints for other developers to use. Overall, this project was a valuable learning experience and gave us the opportunity to apply our knowledge to a real-world project in a collaborative team environment.

# References

|  |  |
| --- | --- |
| [1] | "slack.com," Slack, [Online]. Available: https://slack.com/. |
| [2] | "discord.com," Discord, [Online]. Available: https://discord.com/. |
| [3] | "Microsoft-teams," Microsoft, [Online]. Available: https://www.microsoft.com/en-us/microsoft-teams/group-chat-software. |
| [4] | "github.com," GitHub, [Online]. Available: https://github.com/. |
| [5] | "discord-bot," Discord, [Online]. Available: https://discord.com/developers/docs/intro#bots-and-apps. |
| [6] | "slash-commands," Discord, [Online]. Available: https://support.discord.com/hc/en-us/articles/1500000368501-Slash-Commands-FAQ#:~:text=Slash%20Commands%20are%20the%20new,command%20right%20the%20first%20time.. |
| [7] | "slack.github," GitHub Inc., [Online]. Available: https://slack.github.com/. |
| [8] | "pipedream," [Online]. Available: https://pipedream.com/apps/github/integrations/discord-webhook. |
| [9] | GitHub, "Authenticating as a GitHub App installation," [Online]. Available: https://docs.github.com/en/apps/creating-github-apps/authenticating-with-a-github-app/authenticating-as-a-github-app-installation. |

Conference Name:ACM Woodstock conference

Conference Short Name:WOODSTOCK’18

Conference Location:El Paso, Texas USA

ISBN:978-1-4503-0000-0/18/06

Year:2018

Date:June

Copyright Year:2018

Copyright Statement:rightsretained

DOI:10.1145/1234567890

RRH: F. Surname et al.

Price:$15.00