RepoRanger: Streamlining Collaboration for Software

Engineering Teams

The Bit Co.

Kanad Naleshwarkar   
 kanadn@vt.edu

Sanket Bhujbal   
 sanketb@vt.edu

Sarthak Banerjee   
 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.

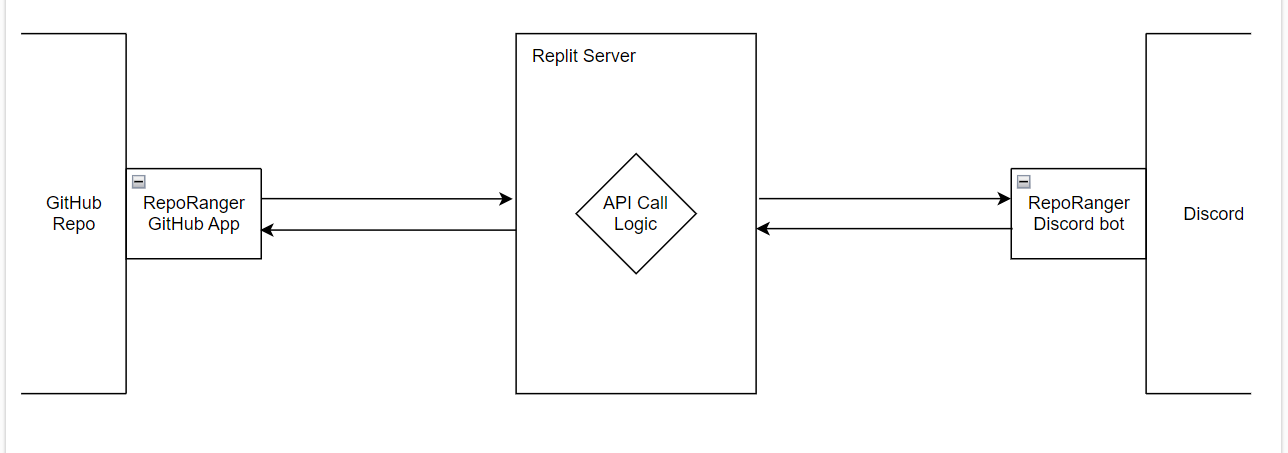
This integration, for instance, can help managers get notified about recent issues, pull requests, etc., or fetch a particular issue from the backlog so that it can be discussed with the team in the same Discord channel. The RepoRanger bot will facilitate this communication by providing a set of slash 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?
2. **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. Based on this inspiration, we have developed RepoRanger. We aim to achieve maximum usability of this bot by making it easy to configure with any GitHub repository and project.

Figure 1: Architecture diagram for RepoRanger

1. **IMPLEMENTATION DESCRIPTION**

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 to collaborate with other teammates. With a shared editor, we could work on different features simultaneously.

1. **DEPLOYMENT PLAN**
2. **MAINTENANCE PLAN**

We aim to maintain our RepoRanger bot by the following means:

1. Regular Updates: The bot implementation will be kept up-to-date with the latest version of dependencies. Doing this will ensure that the bot continues to function as expected and takes advantage of any new features or security patches.
2. Testing: Regular testing will be carried out to account for any change in GitHub or Discord API policies. This will be done to check the bot functionality is not affected.
3. Security Measures: The authentication method we are using to communicate with GitHub APIs is a bearer token. This token will be regenerated and replaced periodically so that it cannot be exploited.
4. Limitations
5. Future Work
6. Conclusion

# 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. |

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