Discord DevOps

A meeting management and kanban ticketing system for Discord

Colin McGee
Intermed Software Design
Virginia Tech
Blacksburg, VA, USA
colinm25@vt.edu

Minwu Kim
Intermed Software Design
Virginia Tech
Blacksburg, VA, USA
kminwu@vt.edu

Jay Doyle
Intermed Software Design
Virginia Tech
Blacksburg, VA, USA
jsndoyle@vt.edu

Ryan Knight
Intermed Software Design
Virginia Tech
Blacksburg, VA, USA
rknight2020@vt.edu

ABSTRACT

In our project, we design and implement a Discord bot that provides tools for formal work meetings and SDLC implementations within Discord. Many software developers use Discord and/or Slack as important parts of their daily work. However, existing tools for SDLC and work meetings are typically external to Discord and Slack, despite both applications having the capacity to house such apparati. Our project provides such capabilities in Discord, extending its utility for engineers and other developers.

INTRODUCTION

Managing a software team is a complex task. Every member must be aware not only of their own contributions to the project, but also the contributions of other members of the team to keep the project flowing smoothly. Without proper communication, team members are more likely to write incorrect code or struggle with issues that would be easily solved in a group. Additionally, keeping team members on track helps everyone stay busy and know what they're working on.

Meetings are an essential part of this management system. However, planning meetings can be unexpectedly difficult, especially when individual team members may be required to attend multiple unrelated meetings that are not shared by others. Because of this difficulty, we intend to take advantage of untapped opportunities lying within Discord's vast software landscape. Discord offers an easy-to-use collaborative chatting environment that can, through its bot functionality, support arbitrary features on top of its platform.

We add two major features to Discord through a Discord bot application. The first is a meeting scheduling system. Discord already provides features that can be used for this purpose, but they are inadequate for software developers working in the kind of fast-paced work environments we see today (we describe further details in the RELATED WORK section). Our bot supports planning meetings around the schedules of individual members.

The second feature is a ticket management system. Ticket management is critical for keeping developers on task by dividing out work and managing issues. Discord does not have an existing analogue for this, so it is an entirely new addition. We offer a Kanban-style ticket management system, where each ticket is divided into different classes based on how far along they are in the pipeline.

BACKGROUND

Discord is a well-known online application for communication. It allows for both text and voice communication, with advanced organization features such as multiple "channels," threads, message pinning, and notifications (including mobile push notifications). Discord originally focused on serving the gaming community, but has expanded to include other groups as well. Today, Discord can be found in many workplaces and schools to facilitate communication in these communities.

In the workplace, Discord often competes with Slack, a similar communication utility. Slack provides more professional tools, such as better group and company management. However, many new developers have more familiarity with Discord from their personal lives, making it a more desirable choice.

Kanban is a development framework used to implement agile processes and DevOps development. It makes use of real-time communication and management of work items on a highly-visible "kanban board," which allows each member of the development team to stay up to date with the state of different parts of the project.

Kanban boards consist of "tickets," wherein each ticket tracks the state of a specific task or piece of work that must be completed. Tickets remain on the board for the entire duration of the project, and are moved between different statuses. These statuses often include a "not started," "in progress," "in review," and "completed" status. Tickets are never moved to a previous status—instead, if a defect is found in a completed ticket, a new ticket is created to address fixes.

Using Kanban, developers are able to keep track of everything happening with a project so that they can focus on implementation instead of management. Additionally, having a shared board helps make sure developers are aware of available work so that they can both keep busy and help others who need assistance. This last point is also aided by daily stand-up meetings, which the Kanban style promotes as part of its Agile focus.

RELATED WORK

Trello [3] boards are an implementation of Kanban boards, created by Atlassian. Trello implements Kanban boards by allowing users to create categories for each ticket and then use a convenient UI to drag and drop tickets between categories. Additionally, Trello supports advanced ticket management and description features, allowing users to keep track of any information they want in the ticket.

However, Trello is detached from developers' main meeting point. Trello is hosted on an external website, not integrated with a communication system like Slack or Discord. Though this could be remedied by an integration (e.g. Discord webhook), it can also be solved by moving ticket tracking into the communication system itself, removing any dependence on an external website. This should improve developer ergonomics, and subsequently, their productivity.

Wright’s writing on using a Discord bot for education during the COVID-19 pandemic demonstrates the potential and value of our idea [4]. They used their bot for various objectives both orthogonal and related to our own problem, demonstrating the potential for using Discord to achieve our goal of convenient ticket management.

In terms of meetings, Discord’s own event-handling capabilities could be said to compete with our bot. However, Discord's native implementation fails to account for varying schedules and managing events on any deeper level than accounting for availability with a group unable to give feedback. Additionally, Discord provides no facilities to prevent double-booking, allowing a user to accidentally RSVP for a meeting that they will not be able to attend.

Our meeting solution fixes these issues in the simplest way possible: by adding support for additional features and checking a user's timeline to make sure that they are available when RSVPing for a meeting. Meeting-planners will then know whether or not their meeting time is viable without having to painstakingly plan a meeting with the group, likely using a separate meeting, which suffers from the same problems, or via mass email, which could lose focus.

IMPLEMENTATION

Our bot is implemented using the Nextcord Python package, which is a fork of the earlier (and perhaps more well-known) Discord.py project. Each feature in the bot makes use of Discord's "slash commands" feature, which makes it easier for users to create and understand the commands that a bot supports.

To create the bot, we used the "prototyping" software engineering process. This allowed us to create smaller-scale variants of the project to narrow down what is specifically most valuable to users, which helped focus development efforts into the areas where it would have the most impact. Additionally, rapid prototyping helped identify bugs early, which resulted in a more useful and stable final product.

Logic within the bot is designed using the "state" model, with event-based triggers (i.e. user commands). This model is useful for keeping track of user activities and reacting to changes that are requested, instead of constantly polling for updates. By keeping state, users are able to peek into the state of the system and modify it with commands.

The bot is divided into two subsystems: the meeting and kanban subsystems. The following sections describe the implementation of each.

The meeting subsystem consists of a group of objects representing meetings and their users, with additional objects for tracking sub-elements (such as time windows). When a meeting is created, the system creates a corresponding meeting object behind the scenes. Users RSVPing to a meeting "join" this meeting by registering themselves in the meeting object, which also marks them as busy during that time. Other commands provide views into these meeting objects so that users can understand the state of different meetings.

The kanban subsystem is composed of a group of ticket objects, with commands to view and modify these tickets. Tickets themselves are entirely stately, with no logic besides requesting different representations (such as embedding to be viewed). Commands directly create and modify these ticket objects, which are stored based on their ticket ID (which never repeats).

DEPLOYMENT PLAN

Bot deployment is relatively straightforward. First, the interested party must create a Discord developers account and create a bot account. Then, load the Discord bot's token into our application. Invite the bot to the Discord guild (server) where it will be used. On the server machine, install the pip packages "nextcord" and "python-dotenv". After that, simply run the bot code using Python 3.8+. The bot will automatically accept commands from any server members.

To support changes necessary, the bot can be modified on a per-server basis. Each component of the bot is self-contained, which improves the ability to fix issues and make desirable changes when they are brought up.

DISCUSSION

There are certain limitations with Discord that also cause challenges for our project. For example, Discord imposes a 10-field limit on embedded objects, so we have implemented a paging system to overcome this. However, it comes at the cost of developer ergonomics (making it more difficult to find certain information), so we do not believe this to be an adequate solution.

Additionally, limitations with how a bot using slash commands is allowed to take action make it difficult to add features such as meeting reminders. For example, bots are not allowed to arbitrarily send messages without increased permissions.

Other possible expansions include the ability to rename meetings and options to list a user's "meeting plan" for the current day (or any other day). In terms of ticket management expansions, we consider the possibility of adding additional fields and more expressive output.

CONCLUSION

This project demonstrated a meeting management and kanban ticketing system implemented using a Discord bot. This system allows for increased developer ergonomics by integrating these commonly-used systems with a developer's normal communication system, reducing the need for reliance on external tools. Future improvements include additional features such as renaming meetings or listing all meetings planned for the day, but the bot's ability to perform basic tasks for meeting scheduling and ticket management prove Discord's feasibility as a solo communication platform for software developers.

REFERENCES

[1] Patricia S. Abril and Robert Plant, 2007. The patent holder's dilemma: Buy, sell, or troll? *Commun. ACM* 50, 1 (Jan, 2007), 36-44. DOI: https://doi.org/10.1145/1188913.1188915.

[2] D. Radigan, "What is kanban?," *Atlassian*, 2022. https://www.atlassian.com/agile/kanban

[3] Trello, "Trello,", *trello*, 2023. https://trello.com/

[4] D. Wright, T. Severance, C. Knutson, J. Krein, and T. Buchanan, An Autonomous Discord Bot to Improve Online Course Experience and Engagement: Lessons Learned Amid the COVID-19 Pandemic. 2022. Accessed: Sep. 22, 2023. [Online].