GSoC 2024

Project proposal for **Rocket.Chat**

Agile Bot

by Hardik Bhatia

Table of contents

Why me?

I started contributing to Rocket.chat as a way to enhance my skills in real world software development, and I have gained an immense amount of knowledge and experience from contributing to this project. I believe my skills in React, TypeScript and AI development that I have honed over the past 2+ years make me the perfect candidate to take up this project.

I have committed the past 4 months to understanding the Rocket.chat codebase and in doing so, I have come to embrace all the nuances that go into making production level software. Some invaluable things that this experience taught me are writing tests, following coding standards, feedback-driven issue resolution, and collaboration. Attending the workshops hosted by the Rocket.chat team has also been extremely helpful in getting started with the development of our own AI apps.

The prospect of contributing to a piece of software that is directly used by over 12 million people really excites me, and if selected, I want to learn all I can from the talented team at Rocket.chat. Doing this project will help me gain valuable knowledge and experience about how software development happens outside of classroom projects and competitions.

2. Project Overview

Abstract

This project aims to enhance team productivity within agile environments by using a chatbot-based approach. By developing a chatbot that is customized to meet the needs of agile workflows, teams will be able to streamline communication, task management, and meeting coordination.

Project duration: 175 hours (Medium)

Project difficulty: Intermediate

Deliverables

Implementation of this project will require extensive use of API endpoints and creation of a bot using the Apps-Engine which will function like Rocket.cat in some ways. The key features of this bot will be:

- Reminder functionality for task deadlines and meetings.
- Providing relevant links and resources.
- Using emoji reactions and similar inputs to carry out tasks like meeting attendance, event confirmation, polls and more.
- Support for daily scrum/kanban for sync and async teams.
- Meeting aggregation and reports.
- Cross platform integration with tools like JIRA.

Additionally, the project will also try to cover these features, but they can also be implemented later down the line as they require heavy feedback over longer periods of time for proper real-world implementation:

- Workload balancing for teams.
- Data analytics for gauging team performance.

3. Project plan

Tasks overview

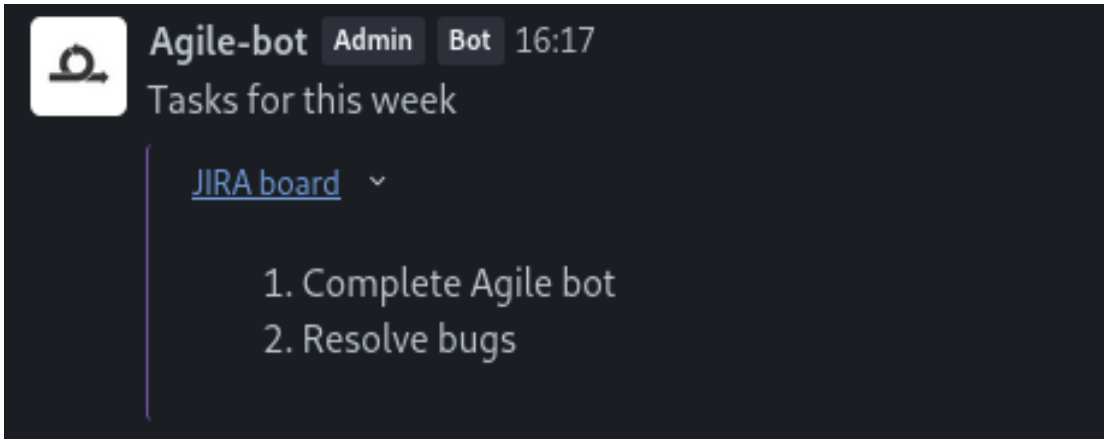
The bot will occupy a single channel consisting of a team. A simple setup process will allow the bot to gain access to different tools which help facilitate the agile framework like JIRA.

1. Weekly tasks reminder

By utilizing the MessagePersistence class in Rocket.chat apps-engine, we can have daily reminders of weekly tasks that are privately messaged to users at the start of the work day.

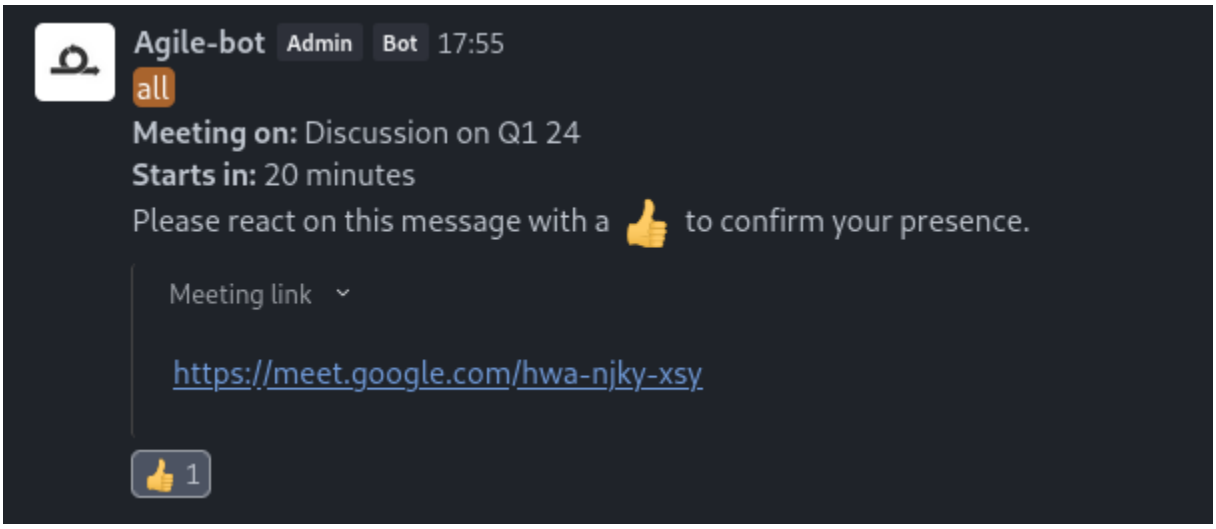
```
    public static async persist(persis: IPersistence, room: IRoom, id: string):
Promise<boolean> {
        const associations: Array<RocketChatAssociationRecord> = [
            new RocketChatAssociationRecord(RocketChatAssociationModel.MISC,
'message'),
            new RocketChatAssociationRecord(RocketChatAssociationModel.ROOM,
room.id),
            new RocketChatAssociationRecord(RocketChatAssociationModel.MISC, id),
        ];
```

This will allow us to reduce the number of API calls to JIRA. This information will be sent to the squad members via personal message (frequency can be changed according to the requirements and guidance of agile experts and mentors)



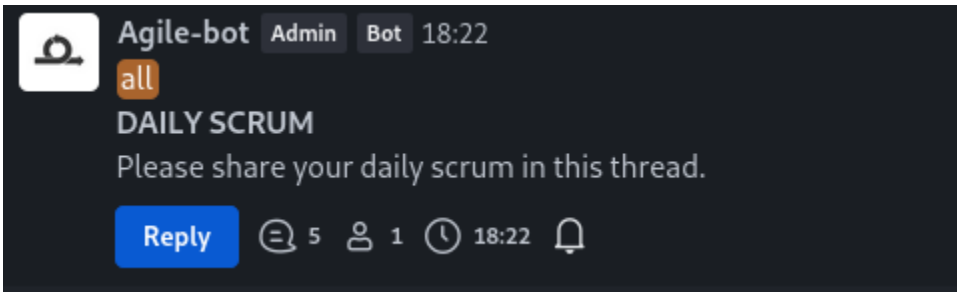
2. Meeting and deadline reminder functionality

By using the various endpoints in ‘packages/rest-typings/src/v1/calendar/index.ts’ along with the Google Calendar API, we can facilitate meeting and deadline reminders for team members.



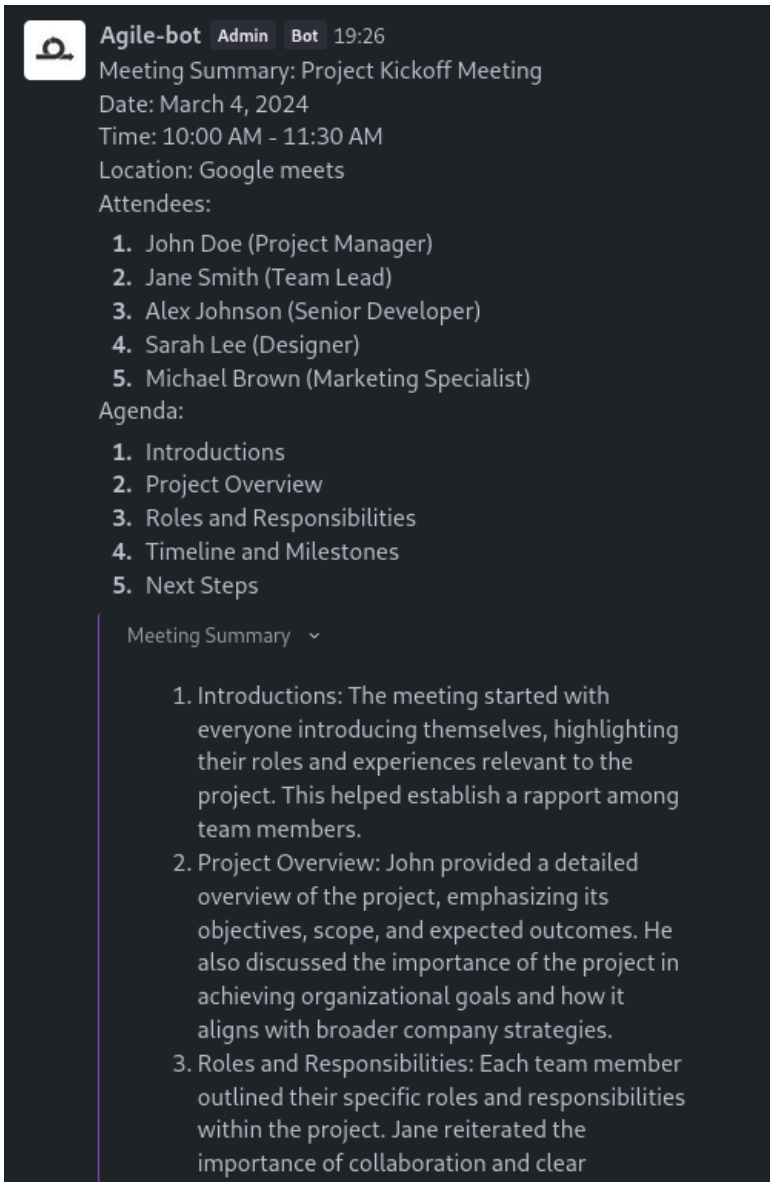
3. Daily scrum facilitator

Using threads, we can aggregate the daily scrum of the team. This can be done asynchronously via reminders that are in accordance with the person’s local timezone (which can be inferred from user’s Rocket.chat settings), or in case of synchronous teams, UTC can be used. A common thread can be utilized for members to post their scrum, which is stored and shared at a fixed time. This component can benefit from utilization of AI to summarize the daily progress at a fixed time.



4. Meeting summarization and reports

After meetings end, a summary of the meeting will be posted by the bot, and tasks will be presented. AI will play a critical role in accomplishing this task. Meeting transcriptions will be used to accomplish this.



To leverage the AI capabilities of Rocket.chat, we'll need to interact with their large language models (LLMs) hosted on the server. This can be achieved through basic API calls. An API, or Application Programming Interface, acts as a messenger between our application and the LLM. We'll send instructions and data through the API, and it will retrieve the LLM's response.

```
async function generateResponse(userMessage: string, prompt: string):
Promise<string> {
    const apiKey = 'YOUR_API_KEY';
    const endpoint = 'https://api.openai.com/v1/completions';

    const requestBody = {
        model: 'text-davinci-003', // Model ID for GPT-3.5 (DaVinci)
        prompt: `${userMessage}\n${prompt}`,
        max_tokens: 100,
        temperature: 0.7,
        n: 1
    };

    const requestOptions = {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'Authorization': `Bearer ${apiKey}`
        },
        body: JSON.stringify(requestBody)
    };

    try {
        const response = await fetch(endpoint, requestOptions);
        const responseData = await response.json();
        return responseData.choices[0].text.trim();
    } catch (error) {
        console.error('Error:', error);
        return 'Error generating response.';
    }
}
```

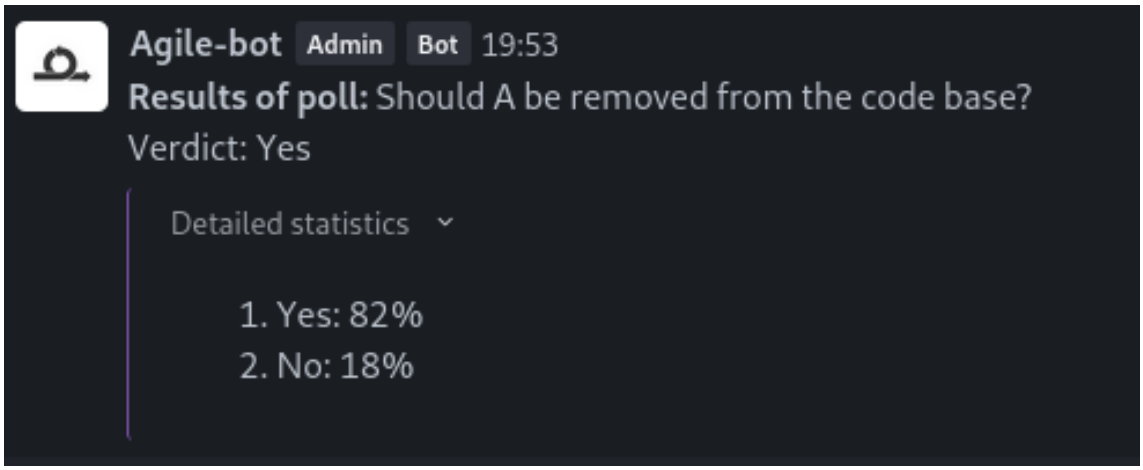
While the provided code demonstrates a call to OpenAI's ChatGPT-3.5, the core concept remains applicable. The specific API structure and method for identifying the LLM will likely differ for Rocket.Chat. Regardless of these variations, the fundamental process will involve making an API call to retrieve a response from the Rocket.chat LLM based on user input and configuration details.

5. Polls and live voting

Polls and voting is an essential part of team meetings, and is best done on the communication platform itself. Timed polls will be implemented using slash commands.

```
export class StartPoll implements ISlashCommand {
  public command = 'poll';
  public i18nParamsExample: string = 'polls_example';
  public i18nDescription: string = 'polls_description';
  public providesPreview: boolean = false;

   public async executor(context: SlashCommandContext, read: IRead, modify: IModify, http:
IHttp, persis: IPersistence): Promise<void> {
      const user = context.getSender();
      const params = context.getArguments();
      const room: IRoom = context.getRoom();

      if (!params || params.length == 0) {
                return this.notifyMessage(room, read, user, "At least one time argument is
mandatory. A second argument can be passed as question.");
      }

      let time = params[0];
      let question = params.length > 1 ? params.slice(1).join(' ') : '';

      await startPoll(user, room, question, time);
      await this.notifyMessage(room, read, user, "Poll has been started for " + time);
  }

   private async notifyMessage(room: IRoom, read: IRead, sender: IUser, message: string):
Promise<void> {
      const notifier = read.getNotifier();
      const messageBuilder = notifier.getMessageBuilder();
      messageBuilder.setText(message);
      messageBuilder.setRoom(room);
      return notifier.notifyUser(sender, messageBuilder.getMessage());
   }
}
```

(Sample slash command implementation for a simple yes or no poll)



Schedule of deliverables

Dates	Tasks
1 May - 26 May	Community bonding period. Get an in-depth understanding of the agile framework by observing real-world scenarios through interaction with mentors.
27 May – 2 June	Develop the first prototype capable of taking inputs via reactions on messages and extracting information from threads.
2 June - 9 June	Add support for data exchange with Jira using their API. Add support for Google Calendar for scheduled meetings and events.
9 June – 16 June	Document, debug and test all the implemented features.
16 June – 29 June	Add functionality for daily scrum, and define a set up process for the bot to collect data and gain access.
1 July – 8 July	Buffer period for debugging, testing, documenting and adding any additional features.
8 July – 12 July	Mid-term evaluation of the project. Reflect upon the progress so far, make changes to the plan if necessary.

13 July – 27 July	Implementation of AI features like meeting summary, scrum reports.
27 July – 3 August	Add polling functionality and prepare the app for publishing on Rocket.Chat marketplace.
4 August – 15 August	Final buffer and testing period for all features.

Time commitments

I confirm that I have no other obligations during the GSoC period from 17th May to 15th August and I will be working upon this project for a minimum of 40 hours every week.

I have exams from 6th May - 17th May (during the community bonding period) so for that duration I will be available for ~25 hours every week.

4. Progress monitoring

To effectively track my project's progress, I plan on maintaining a daily work log and actively participating in daily scrum meetings. These practices will help me stay organized, keep track of tasks completed, and identify any hurdles that may arise along the way. All of this will ensure that I have a clear understanding of my targets and can address any challenges in a timely manner.

In addition to this, I've decided to share my journey through bi-weekly blogs on Medium. These blog posts serve as a platform to reflect on my experiences, share my learnings, and connect with a wider audience. By documenting my progress, I will contribute to the knowledge within the community and also reinforce my own learning and growth throughout the project.

5. Future scope

The Agile Bot will be an app in the Rocket.chat marketplace. The use and scope for this project goes beyond a single GSoC project. If granted the opportunity, I will continue to maintain and develop this project even after the GSoC duration as it is something I feel deeply passionate about and can be improved and built upon by the community.

6. Benefits to the community

The Agile Bot will be a helpful assistant for the Rocket.Chat users, making teamwork smoother and more efficient. It will keep everyone in the loop with updates and reminders, and it takes care of repetitive tasks so team members can focus on what's important. By fitting well with agile methods, it will encourage collaboration and adaptability, which are crucial for successful projects within the community.

For companies, Agile Bot will be helpful for managing projects. It will help teams plan their work, manage tasks effectively, and stay on track. Features like meeting and scrum summaries and polling will make Rokcet.Chat an integral part of the project process, thereby increasing the productivity and efficiency of the Agile process. Plus, it will work alongside other tools companies already use, making it a perfect fit for any workflow.

7. Conclusion

The agile bot is an extremely intricate and feedback-driven project which will require constant change in approach and implementation to be a success. Due to this project’s highly unique nature, the role of the mentors in providing guidance about the agile ecosystem will be of paramount importance to the completion of this project.

Finally, I want to express my gratitude to the Rocket.chat staff for considering this proposal and for helping me grow and mature as a software developer. Every interaction with the maintainers and the community has taught me a valuable lesson which will help me immensely in my career for years to come and I will continue to contribute to and be a part of this amazing community for a long time.