FocusBot

ANDREW JELSON*, SHEAN KIM, and SHREYAS PAWAR, Virginia Tech, USA

CCS Concepts: • **Software and its engineering → Software design engineering**; **Designing software**; *Programming teams*; Software prototyping; *Software development techniques.*

Additional Key Words and Phrases: Software engineering, motivation bot

ACM Reference Format:
Andrew Jelson, Shean Kim, and Shreyas Pawar. 2023. FocusBot. 1, 1 (May 2023), 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 ABSTRACT

This project focuses on building a motivational Slack bot "FocusBot", for remote users to access their work calendars, add work schedules, and get motivated for day-to-day responsibilities. This bot is integrated to slack channels which uses the concepts of Pomodoro techniques and daily quotes to keep the users motivated and efficient. Additionally, the bot also helps track work schedules by integrating tasks with an online Firebase database with separate workflows for both managers as well as users.

2 INTRODUCTION

There were many situations where we had to work from home during the COVID-19 pandemic. Because of this, research has been conducted which shows that working remotely has advantages such as reducing commuting time, fatigue, and reducing the stress of workers through a comfortable working environment so many workplaces are continuing to utilize this methodology[1, 2]. However, if we work remotely, it is easy to lose concentration, so there are cases where work efficiency is rather low [3–5]. Spaces where we work remotely, such as at home, often coexist with spaces for various purposes, such as spaces for relaxation and dining, and these environments make it difficult to focus only on work. In addition, it is difficult to get feedback on work progress and plan daily agendas because there is a high probability that colleagues and bosses are absent in the space, making it difficult to manage the work on your own and easy to cause delays. We identify some major issues with working remotely including being easily influenced by non-work factors, being psychologically inattentive, having difficulty focusing on and making progress with work, and establishing daily plans.

This project aims to increase the efficiency of users by allowing them to focus on daily tasks in an environment that requires remote work. It also tries to prevent delays by helping users easily understand their work progress and track their progress with calendar integration. In addition, by allowing users to share their progress and plans with team members and team leaders, we intend to

*All authors contributed equally to this research.

Authors' address: Andrew Jelson, jelson9854@vt.edu; Shean Kim, shean@vt.edu; Shreyas Pawar, shreyaspawar@vt.edu, Virginia Tech, Blacksburg, Virginia, USA, 24060.

XXXX-XXXX/2023/5-ART $15.00
https://doi.org/XXXXXXX.XXXXXXX

provide an environment where they can check each other's progress and exchange feedback. Lastly, it will help users get into work mindsets by using Pomodoro techniques and motivational quotes.

3 RELATED WORKS

There are several tools that increase work efficiency and help collaboration in remote environments. A typical example is Jira [6]. Jira is one of the project management tools that help manage projects using Agile and supports functions such as planning, assignment, tracking, and reporting to enable collaboration among team members. However, with Jira, it is difficult to know the priorities between tasks without dependencies. Also, there are so many functions that it takes a lot of time to learn how to use them. It is also difficult to apply to projects that manage tasks through other software engineering methodologies because they are only based on Agile. Jira is also a software device, whereas we are developing a bot system.

Slack [7] is another representative collaboration tool and is often used in environments where you need to work remotely because it helps communication between team members. However, because Slack itself does not have other features related to the project, such as checking the progress of the task, planning and motivating users to focus on their work, other tools such as Trello [8] and Google Calendar [9] are used in conjunction with Slack. However, these tools are not fully integrated with Slack, which means that tools often require access through separate apps or web pages. The use of Slack and various tools in conjunction helps plan and manage work but does not support the function of creating an environment where users can focus on work in a remote environment. FocusBot is an extension that will implement the features of Trello over Slack so users won't have to switch platforms like with Jira.

As we developed our system, we found the need to have a database to store user keys and employee types, so we started using Firebase [10] to store this data and create the calendar. We moved away from Google Calendar implementation because it is more efficient to use a database for the manager view.

In addition, there are several tools to help remote working environments such as SmartSheet [11], OnBoard [12] and Asana [13], but they do not provide integrated functions such as tasks management and creating an environment where users can concentrate on their work, or require the purchase of expensive plans for integrated functions.

4 MOTIVATION

The isolation during Covid was unexpected and has brought a lot of changes to the personal and professional lives of people. Working remotely had various advantages, but also made it difficult to focus and concentrate on the work, given the distractions that exist working from home. These distractions often make users lose their motivation to work alone thus making them less efficient. This project aims on helping improve this problem, especially considering the adoption of the hybrid work models of software companies, where they allow users to work from home for some days in today's times.

For instance, using FocusBot, the user is greeted with a motivational quote every time they start their workday. Furthermore, the users will also be able to keep track of their schedule using the schedule integration feature of the FocusBot which will provide a list of the projects and tasks that are required to be worked upon by the users. The schedule feature also promotes collaboration among teams as the tasks would be divided based on different projects the users work on along with their managers. The FocusBot also implements the Pomodoro [14] techniques of tracking the user's productivity and allowing the user to take breaks which have proven to help improve efficiency.

5 PROCESSES

This project used a layered architecture pattern to structure the system because it relies on users from different categories (users, managers) which could potentially work with multiple devices, all interacting with a database. The system can be broken down into two major components - a managerial component and a user component.

Both components will have access to the database, where the user component will have access to specific user records, while the management component will have access to all users within a project. The user component will also have their own deadlines and tasks visible to them, while the managerial component will have all the tasks of a particular project along with the users visible to them.

There will, however, be a clear distinction between the access rights of each component. The management component will only see the tasks that are related to a project for a particular user instead of all the users' tasks. Similarly, the user will only be to view individual tasks and/or the tasks that require collaboration between the user and someone else. These features will be implemented using a deadline-based architecture.

Other key features of our design include the Pomodoro technique and a motivational daily message. The user will give the bot a time of day, and the bot will send out a daily motivational quote along with the key tasks that the user has in the near future (based on the above calendar system). It will then use Pomodoro techniques of 30 minutes of work and a 5-minute break to give the user a good work schedule.

This project also followed an iterative development model where we used a series of prototypes to create our final system. With each iteration, we focused on implementing a new feature. In our first model, we created a bot that can communicate with Google Calendar and has both add and remove features for different tasks. We then added the motivational quote database to our codebase allowing our bot to send motivational quotes on demand. From there we decided to switch from Google Calendar to Firebase for task tracking because we needed the database to store different users. After these different versions of our bot were created, we added in the Pomodoro techniques and the daily message feature. This allows our bot to send daily motivational quotes and track tasks for each user. In our last iteration, we create different user levels, like employee or manager. Managers can view all employees' task tracking and see how each member has worked on the different features, and when they will likely finish their sections.

6 IMPLEMENTATION

After looking into Slack's API software, we found that Python has a lot of packages for creating bots. Because of this, we chose Python as our programming language. We also chose to use Visual Studio Code as our IDE because we have all used it in the past and it has helpful features such as GitHub integration. For our version control, we chose to use GitHub as it is the industry standard, and we have all used it for projects in the past. We created a few branches so we wouldn't interfere with each other's coding, and we merged regularly to integrate new features in steps. In terms of code ownership, we went for a collective code ownership model where we each worked together on the same files. Because the bot will ultimately run out of one file, we all edited the same file to keep our code efficient.

We used a variety of different packages to make FocusBot work as intended. The main bot uses the firebase_admin API to communicate with our database, Flask API for hosting the code through the web, and slackeventsapi to communicate directly with Slack. We also used Pandas to create our motivational quote files.

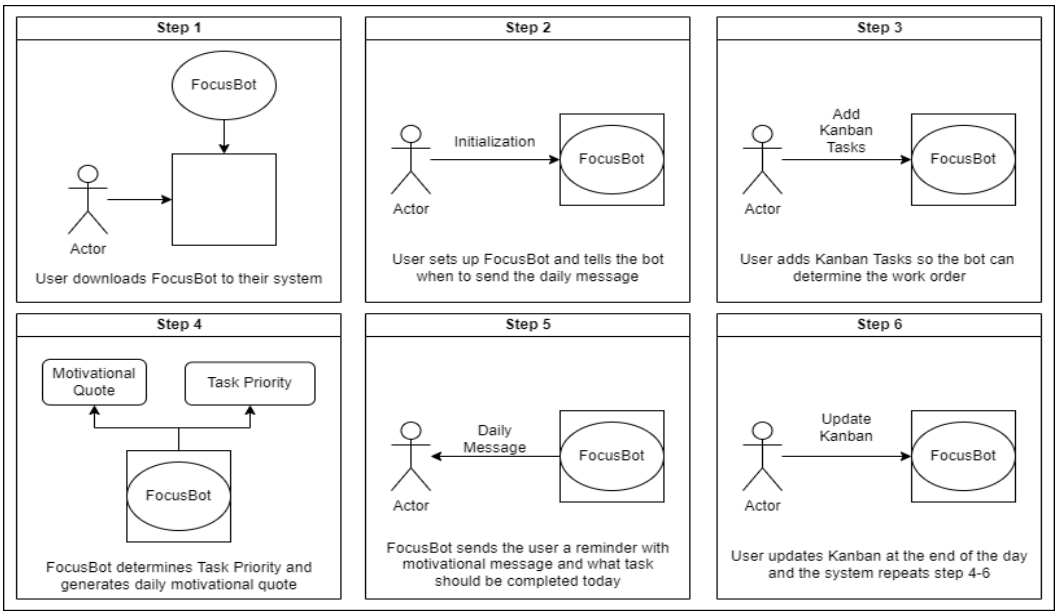

Fig. 1. Storyboard for a potential user of FocusBot

7 TESTING

The project was tested using two pre-defined dummy users with dummy tasks in the initial phases to validate the correct functioning of the Firebase integration. After the complete development of the project, the project was then manually tested using two different users and one manager user with Slack integration to test out the correct functioning of the project.

There was no Slack testing API to be found for Python modules, especially considering that the app was also run on Flask. Hence, for the unit test cases for the project, a separate file of Python tests was developed, which tested on a slightly modified deployment project file, which simulated the same functioning of the original project, but had changes made to it so that it could be tested using unit tests in Python.

8 DEPLOYMENT

8.1 Deployment

To deploy FocusBot using DevOps, we have taken several measures. During the development phase, we built the server locally using ngrok, but for deployment, we will use AWS EC2. Code management is currently being done through Git, and any modifications or additions to the code after deployment will also be done through branching and merging in Git. To ensure a streamlined process, we plan to automate the process of building, testing, and deploying the code using AWS CodePipeline.

For the deployment, we'll be using canary deployment with several phases. The first deployment will target teams of up to 5 members who collaborate using Slack. The subsequent deployments will target teams of up to 10, 20, and 50 members, respectively. Finally, we will deploy to all users who collaborate using Slack. After each deployment, we will monitor user feedback and the service's status to make bug fixes and modify FocusBot's functionality accordingly. If we determine that no further bug fixes or feature modifications are necessary after the first deployment, we will proceed

with the second deployment, repeating the previous steps. The remaining deployments will follow the same process.

For testing, we will use fuzz testing. If AWS CodePipeline has built-in fuzz testing available, we will use it for initial testing. We will continue to explore whether there are more suitable automated fuzzing techniques for testing FocusBot, and if such techniques exist, we plan to utilize them.

8.2 Maintenance

We follow an iterative software process to develop our projects. We will also use this process after deployment for maintenance. That means we plan to iteratively test and modify our code to ensure maintainability. Additionally, we use Floss Refactoring to improve the maintainability and readability of the code, as well as to address security vulnerabilities. This approach will allow us to continuously improve the quality and security of FocusBot.

To further ensure a high level of service quality, we will continuously monitor the service in real-time using AWS CloudWatch. This allows us to quickly respond to any bugs or issues that arise. We will also use AWS Security Hub to detect and respond to any security-related issues promptly. After the final deployment, we will determine the maintenance and upgrade schedule based on real-time service status monitoring and user feedback.

9 FUTURE WORK

To improve and add more functionalities to the working of the FocusBot several extensions could be made. However, those would not be viable to implement within the scope of this project but could be implemented in the future if time permitted:

- **UI Improvement:** The tasks to be added to the database uses a particular format message required to be given by the user. This could be made better by integrating an interactive UI with Slack, which will better help navigate around the app.
- **Personalized messages:** Currently, the FocusBot provides the same prompts and messages to all users. In the future, it could be adapted to provide personalized prompts based on individual work styles, priorities, and habits.
- **Gamification:** Some users may respond well to a gamified approach to productivity. The FocusBot could incorporate game-like elements such as points, badges, and leaderboards to motivate users and encourage them to stay on track and compete with fellow coworkers.
- **Natural language processing:** Currently, the FocusBot likely relies on pre-determined prompts and responses which are hardcoded in our SlackAPI handling. In the future, it could be enhanced with natural language processing capabilities that enable it to understand and respond to users' requests in a more conversational manner.

10 LIMITATIONS

The main limitation that we have is that we don't have our code permanently hosted on the web. For deployment, we would use the AWS EC2 cloud computing, but we don't have the funds for it. We looked into the cloud.cs.vt.edu system, but we were not able to get our code hosted at this time. Another limitation of our project is that we have to hardcode the managers into the system. We decided that having a set_user function would not be viable because we don't want users changing status at will. Lastly, our system user interface requires specific instructions on how to add and remove tasks. In the future, we plan to improve upon the UI and have an alternate method for setting managers in the bot's database.

11 CONCLUSION

FocusBot helps manage schedules easily and increase work efficiency and focus while using Slack. FocusBot provides an interface that allows team members to easily check their to-do list via Slack commands, and add/delete tasks from their schedule using Slack commands as well. It also assists team managers in checking and managing the schedules of their team members easily. Additionally, we have added the Pomodoro feature to FocusBot to help team members increase work efficiency by balancing concentration and rest time, and to motivate team members with daily quotes. FocusBot is especially helpful in situations where team members need to work from home, as it clarifies individual tasks and goals and motivates team members to perform their work efficiently without losing focus.

REFERENCES

[1] S. Leroy, A. M. Schmidt, and N. Madjar, "Working from home during COVID-19: A study of the interruption landscape," in *Journal of Applied Psychology, 106(10), 1448–1465*, 2021.
[2] D. Ker, P. Montagnier, and V. Spiezia, "Measuring telework in the COVID-19 pandemic," in *OECD Digital Economy Papers, No. 314*, 2021.
[3] Y. Shao, Y. Fang, M. Wang, C.-H. Chang, and L. Wang, "Making daily decisions to work from home or to work in the office: The impacts of daily work- and COVID-related stressors on next-day work location," in *Journal of Applied Psychology, 106(6), 825-838*, 2021.
[4] S. Chong, Y. Huang, and C.-H. Chang, "Supporting interdependent telework employees: A moderated-mediation model linking daily COVID-19 task setbacks to next-day work withdrawal," in *Journal of Applied Psychology, 105(12), 1408-1422*, 2020.
[5] C. Andrade and E. Petiz Lousã, "Telework and Conflict during COVID-19 Lockdown in Portugal: The Influence of Job-Related Factors," in *Administrative Sciences, 11(3), 1-14*, 2021.
[6] Jira, "Jira," https://www.atlassian.com/software/jira.
[7] slack, "Slack," https://slack.com/.
[8] trello, "Trello," https://trello.com/.
[9] google-calendar, "Google calendar," https://calendar.google.com/.
[10] Google, "Firebase," https://firebase.google.com/.
[11] SmartSheet, "Smartsheet," https://www.smartsheet.com/.
[12] onboard, "Onboard," https://www.onboardmeetings.com.
[13] asana, "Asana," https://asana.com/.
[14] Todoist, "The pomodoro technique," https://todoist.com/productivity-methods/pomodoro-technique.