The Motivation Bot:

Andy Nguyen, anguyen4@stevens.edu
Erik Buczek, ebuczek@stevens.edu
Dominic Ortiz, dortiz@stevens.edu
Written: May 4th, 2020

I pledge my honor that I have abided by the Stevens Honor System

Signed:

Erik Buczek	Date: 5/4/20
Andy Nguyen	Date: 5/4/20
Dominic Ortiz	Date: 5/4/20

https://github.com/ebuczek525/SSW-345-Bot-Project

https://drive.google.com/file/d/1qI2feGtl3jNNLJ8aGnqgeGjxDfJv2-Pf/view

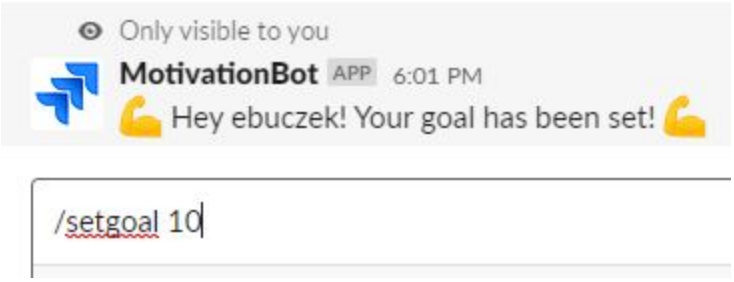
Problem

The main focus of our bot is to address the problem of poor worker productivity, which exists in some cases. We wanted to take advantage of the fact that many workplaces have applications such as Slack and Jira integrated into their projects. Here, we saw the perfect opportunity to create a new bot that would both address the aforementioned issue, while also making use of existing technologies. Our bot, MotivationBot, is intended to make workers more productive by both encouraging and rewarding productivity through a communication-based worker community. Although many companies make use of communication applications such as Slack, many do not utilize them to their full potential. Worker productivity can be a very detrimental issue when it is at low levels, as an unproductive workplace can lead to a loss in revenue and growth. Companies and projects want workers who are motivated and work efficiently, however, many workers feel disinterested in working and can get distracted. Workers can be productive and motivated by being rewarded through gamifying their experience while helping growth and progress on projects and in companies.

Primary Features

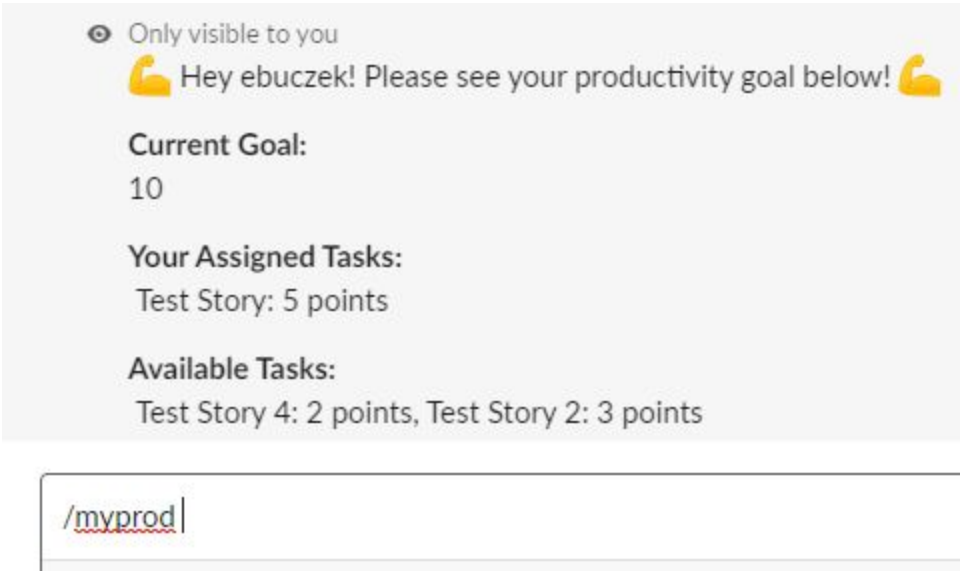
The primary features of the bot include: setting a productivity goal, checking a productivity goal, requesting a task reminder, and “completing” a task.

Setting a goal:
Command /setgoal <Number>



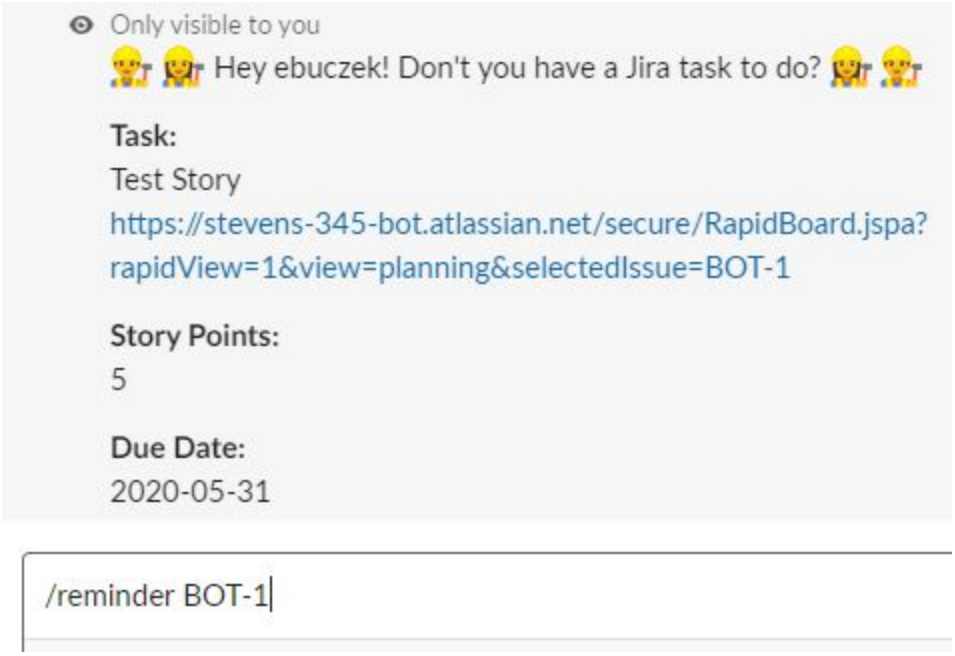
I set myself a goal of 10 user/story points.

Checking a productivity goal:
Command /myprod



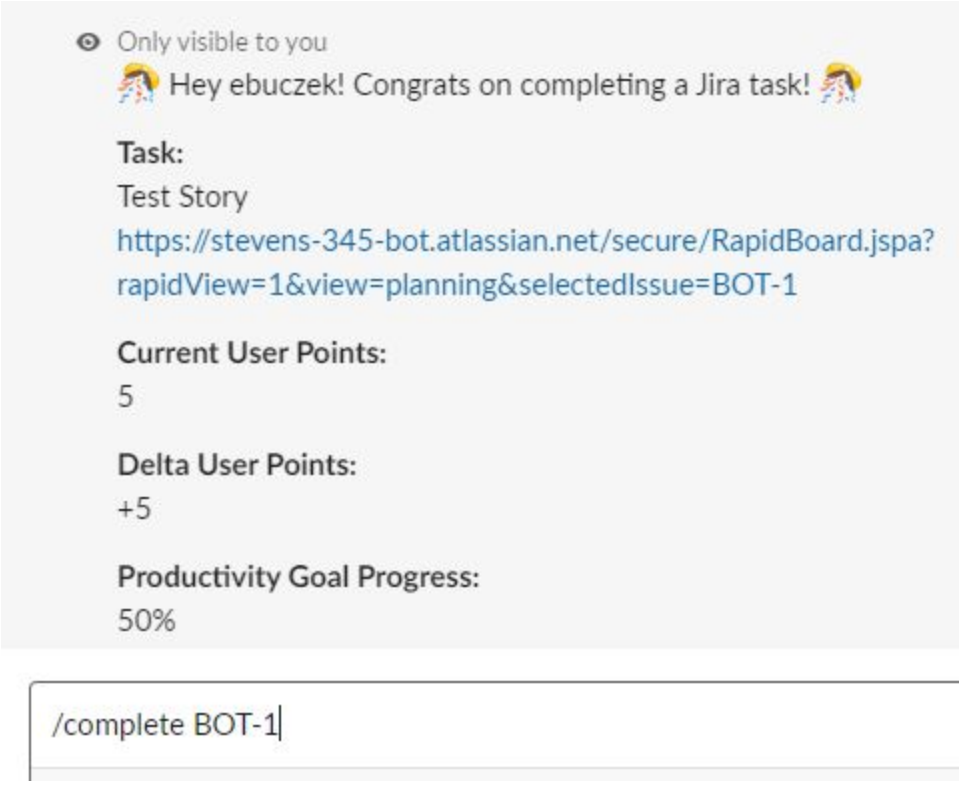
I check my goal and see tasks that are assigned to me and others that are open.

Requesting a task reminder:
Command /reminder <Issue ID>



I can see the following information for an issue: Name, link, story points, and due date.

Completing a task:
Command /complete <Issue ID>



I "complete" this task and see my current user points and productivity goal progress.

Reflection

Initially, the project started off a little slowly, but this was due to the nature of the milestones that were due periodically. For the first week or so, our process had been fairly simple as we had not yet started the code implementation of the project yet. We had simply been documenting our project management process (using GitHub) through the process document in our repo. We included screenshots of our Kanban board progress, as well as some meeting notes and iteration progress. We also created workspaces for Slack and Jira which we will use in various parts of our project implementation. Next, we moved on to implementing the building blocks of our bot. The bot was created in Slack and our Slack webhook was made ready to be used for future development. Also, we created our first slash command, which was used to handle the "Add a productivity goal" feature. We followed some Slack API tutorials in order to create a server as well as a secure tunnel for the localhost, using ngrok. At this point, we created some separate MongoDB files, which we did not end up using in our final

implementation. Instead, we ended up using MongoDB/Mongoose code in our main application file. Next, we had to implement the integration with Jira. We were fortunate to find a node module that enabled us to more easily interact with our Jira board.

The remaining portion of the project involved making API calls using Postman, in order to figure out which data we wanted to use. From there, we kept iterating over our code and adding different functionalities until we were confident that we had fulfilled the promises made in our use cases. Our method of implementation relied heavily on pair programming. All of the development occurred over Zoom calls, in which team members contributed to one living file together. This method may not be efficient in general, but we found that it suited our needs, especially since our server and database were being hosted locally, which rendered asynchronous programming unfeasible. Most of the development went along smoothly, as team members were supportive of one another. However, we did encounter some issues that required a good deal of debugging, these issues were mostly related to manipulating JSON objects that we were getting from our Jira board. Overall, the team is very satisfied with the method of development as well as the final implementation of the bot.

Limitations and Future Work

Some limitations we faced were learning how to properly use the Slack and JIRA apis, only having a local database, and not knowing how to move on with further deployment. The team had initial difficulty learning how to effectively use the Slack and JIRA REST APIs, but this was mitigated with the incredibly thorough documentation with examples and suggestions for the Slack API and the introduction of the JIRA connector module which had very extensive and easy to comprehend documentation as well. Once we found the JIRA connector module we were able to rapidly develop the core functions of our bot. Another limitation would be our use of a local database, this worked incredibly well for our own testing purposes and was very easy to use thanks to our use of Mongoose, but is not sufficient for further deployment of our bot as a user currently needs to have mongoDB installed and know how it works. A hosted database must be created to move forward. Our final limitation is our lack of knowledge about networking and what steps are needed to move onto a final deployment. Building bots was completely new to all of us at the beginning of this project and we are continuing to learn as work is completed.

There is a lot of possible future work for this bot with the vast amount of options the JIRA API has to offer. We would like to expand current capabilities to assigning tasks, editing or commenting tasks, and getting the change log of the tasks as well for direct next steps. After that there are plenty of JIRA interactions that we could

implement after. We also want to expand the current command features especially /remind and /complete. For /remind we would like to send periodic reminders about tasks until their due date with some user customizability on how frequent these reminders come. For /complete we would like to actually close the issues on JIRA with the slash command so the user does not have to go into JIRA to close it themselves. Besides expanding slash commands we would like to develop more robust code for error handling as this is lacking currently. This bot could also be expanded to other popular messaging platforms such as Discord which would allow the bot to reach even more potential users.