Slack-2-Kanban

Team: Slack'in Off

Authors:

Tyler Kim
tdkim@vt.edu

Tyler Buxton
tbux@vt.edu

Patrick Johnson
patrick91@vt.edu

Hunter Jaques-Pownall
hunjaq@vt.edu

Krishna Nair
krishnanair@vt.edu

Abstract:

This paper investigates the integration of an AI chatbot into the messaging platform Slack, creating a user-friendly solution for updating and managing Kanban boards. We look into the practical uses of this product, discussing its key features, technical framework, and the benefits it offers to project teams. We aim to showcase how this integration simplifies task management, allowing for efficient collaboration within the familiar environment of Slack. Real-world examples highlight the practical implications of this approach, emphasizing its potential to enhance team productivity and workflows. The future of workplace technology, where the fusion of AI, Kanban, and Slack work to redefine the way teams work together.

Introduction:

The need for efficient communication and task management has become paramount since organizations have adopted a hybrid work style. As organizations look for increased productivity and efficiency, the integration of artificial intelligence (AI) technologies are slowly getting implemented to address this. This paper explores the integration of an AI chatbot designed to update a Kanban board through the messaging platform, Slack.

A Kanban board allows teams to visualize work and processes, and improve collaboration. At the same time, Slack has become a popular communication platform for teams, focused on real-time conversations and collaboration. Recognizing the strengths of these two tools, the integration of an AI chatbot works with the strengths of these two tools, communication and task execution, increasing productivity.

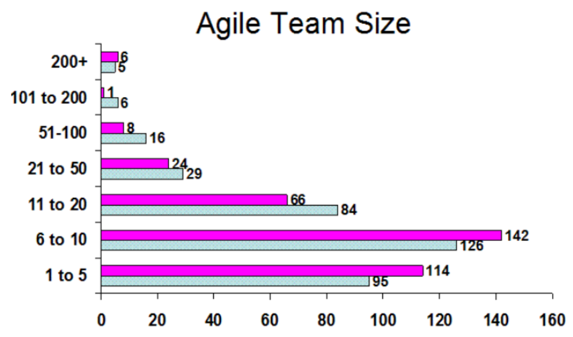
The project focused on the development, functionalities, and impact of the AI chatbot, by enabling users to update and manage their Kanban boards directly within Slack platform. The integration promises to improve task management, offering a user-friendly and efficient way for teams to collaborate, communicate, and maintain their project workflows efficiently. This paper will provide an overview of the AI chatbot's key features, the technical architecture for its functionality, and the potential benefits it brings to project teams.

Additionally some real-world case studies illustrate the practical implications of this integration. It becomes clear that the use of AI for Kanban and Slack really enhances team efficiency and collaboration. By utilizing AI this not only simplifies the task management process but allows teams to focus on creativity, problem-solving, and strategic decision-making.

Related Work:

Our idea for a Slack-oriented chatbot is strongly influenced by the Agile Development cycle. When designing this project, we referenced the article Requirements Engineering in Agile Software Development by Andrea de Lucia and Abdallah Qusef. This article is important for a few reasons. It starts off by defining the agile development process and talking about the pros and cons of it. The pros include quick delivery on developments and the ability to adapt quickly when needed. However, this comes with downsides as well, such as the potential for an overwhelming amount of changes, too many needs to be met, and a lack of a bigger-picture vision. The article then goes on to explain what all should be done in order for the agile process to function effectively, such as establishing smaller sized teams, doing releases frequently, not documenting too early on, and ensuring traceability of requirements.

First the paper establishes that smaller size teams are better for functionality. The figure below highlights the ideal team size for an Agile team.



We also tried to follow frequent releases with our team. This includes frequently delivering parts of the project to meet our milestones. This results in the requirements being completed incrementally and iteratively. Lastly, we focused on the traceability of requirements ensuring that we have well documented milestones.

This was useful to us as it gave us an idea as to 1) if we should use agile in the first place, and 2) what all we should do to ensure that agile works for us. We ended up going with agile because this paper highlighted the strengths of the process, which we were able to take advantage of.

The other related work we referenced was the Jira Software. This is an agile testing and project management tool, which also supports kanban and scrum. Immediately, we can see that this software is very similar to our goal with the Slack bot, such as easing processes and promoting efficiency in work.

This was useful to us as it gave an idea as to where we want to take our project. We wanted to achieve the goal that Jira does, but in our own way without blatantly taking from Jira. Jira gave us ideas on both what we should focus our efforts on, and what we could do without.

High Level Design:

When first contemplating the design this bot should have, we had to consider how the user would interact with the bot, and which structure or pattern would best suit this interaction. Based on this, we planned on an event-based approach. This made the most sense to us as the bot would be reactionary and its actions dependent on what command the user entered. However, the usage is not dissimilar to that of a client-server interaction, where the user is the command issuer and the server is the kanban board. In this train of thought, the user simply manipulates the server's data through the bot, meaning the bot is more of a middle-man and a tool to interact with multiple programs at once. Due to the fact that the user would be only interacting with the kanban data through the bot's method calls, it was also decided upon to focus more on a non-data-centric approach, since only one class (the bot) would need to interact with the data directly.

Implementation:

Since we used Agile in our development of our Slack bot, we were able to implement features in a way that worked nicely for us. Since we didn't have any customers for this product that would request features, we ourselves acted as the customers, adding features that we would find useful if we were to use the product.

Every few days, we would do a scrum meeting where we would update each other on what we were working on and what all needs to be done next. This is another way in which we utilized agile.

As for testing, we thought of all of the uses of the slack bot. We thought of what we want the bot to do, and what the expected output of it should be. We then thought of edge cases (such as assuming an input is there) and tested to ensure that those cases were covered.

Testing:

For testing we need to look at both the backend and frontend of the Bot, because on the backend there needs to be some assurance that the data is fetched correctly. On the frontend that it's clear to the users and that the information is outputted to the correct place.

For the backend ensuring there is proper access to the Kanban API and the Slack API (which was found out on the implementation). Once the backend has access to both of these making sure that the backend can fetch the users (the team members) and the valid tasks that are on the Kanban board.

For the frontend, it is more for the user experience, making sure that it's clear for the user. It outputs the correct name of the developer searched

Maintenance:

Maintenance of the system involves staying proactive in response to updates from the Kanban board API or Slack API. We'll address any changes in data fetching or retrieval methods to ensure the continued operation of the integrated system between the Slack API and the ChatBot. This may involve minor fixes or adjustments to maintain compatibility and reliability.

Beyond technical adjustments, our maintenance plan includes the continued upgrades to new features. For instance, enhancing capabilities to support larger organizations, making the system more scalable. We're also considering options to extend compatibility to other communication platforms like Discord or Teams and considering integration with team management tools such as Jira to broaden the system's usefulness.

Regular checks and updates will be conducted on a rotating basis to keep the system up to date with evolving user needs. This approach to maintenance ensures that the system remains efficient, adaptable, and capable of meeting the requirements of users and organizations. Continuous communication with user feedback and staying connected to industry changes will guide our maintenance strategy, allowing the system to grow alongside technological advancements and user preferences.

Deployment:

Deploying the project involves several key stages, including building, continuous integration/continuous deployment (CI/CD), and test automation. Each stage plays a crucial role in ensuring the code meets the reliability, functionality, and performance before reaching production.

The building stage for our project will involve compiling the source code and creating an executable version of the bot. Then we will focus on the CI/CD aspect. Continuous Integration (CI) and Continuous Deployment (CD) are important to modern software development. CI ensures that changes made to the codebase are automatically integrated and tested in a shared repository. This process helps identify integration issues early. Continuous Deployment, on the other hand, automates the deployment process, allowing for frequent and reliable releases. This was important to following the agile practices outlined in our related work. Integrating CI/CD pipelines with git version control ensures that every code commit triggers automated builds, tests, and deployments. Lastly, test automation is a critical component of the deployment pipeline. Automated tests, including unit tests, integration tests, and end-to-end tests, help catch bugs and regressions early in the development process. In the case of our project, the tests would include executing the commands and validating responses and checking interactions with the Slack/Discord API.

Extensions and Limitations:

If we were to continue to develop the bot, there are a couple extensions we have considered. First we could extend this to be a usable bot for other message boards outside of Slack, such as Discord. Second, we could make it more customizable, to have users be able to fine-tune the interactions and confirmations they get from the bot to better tailor to the user.

As for limitations, we are limited by several factors. We have limited time to fully develop the project as of now, and would be unable to continuously maintain the project to successfully support a large number of users. Additionally, we are limited by how the bot could successfully interact with the kanban board if it were on something such as Github. We currently do not know if sites such as Github would actually be able to be interacted with by the bot to do anything other than have the bot read from the kanban.This is the biggest limitation as of this report, and in order to develop a working product we would need the bot to be able to successfully interact with the kanban.

Conclusion:

The changes and improvements we have made during this project have both developed and adjusted our views on the software development process. In the process of developing an AI chatbot, designed to be implemented into Slack, to create an effective updating and managing tool for Kanban boards.

Through extensive research, we developed our chatbot, using a research article that highlighted the Agile process, for our requirements gathering and analysis. We also based some of our designs and features off software called, "Jira". We further designed our chatbot, using brainstorming techniques, to finalize some of our project attributes.

We then went on to further develop our chatbot, while conducting scrum meetings, and stand ups. Picturing ourselves as the customers, we were able to effectively implement more features into our design. We then conducted our testing, based on our imagining of being part of the customers' perspective. This would continue with fairly routine maintenance, that would be a more continuous approach on updating and developing our product. However, it wasn't without faults, as we had to adjust the platform we were implementing it on in the end, due to some unforeseen disruptions.

We decided to work in a system, focusing on Continuous Integration and Continuous Deployment. We decided that this would be our best option for further development and deployment.

Finally, we decided on changes that we would make if we were to continue to develop the chatbot. This would include extensions into other message boards, and assisting factors like autofill and other similar features.

Overall, we designed a chatbot to improve the flow and order of software development, by taking advantage of the techniques we learned in class. While not finishing at the exact point that we originally anticipated, we were still able to create a functioning chatbot that implemented multiple features that we had desired, while creating a plan for what could be changed or done in the future, given more time.

References:

1. Tate, Thayer. "Software Development Timeline: How Long Does It Take to Build Custom Software?" *SOLTECH*, 13 Mar. 2023, soltech.net/how-long-does-it-take-to-build-custom-software/#:~:text=Planning%20is%20a%20short%20activity,how%20they%20will%20work%20together.
2. Glover, Ellen. "19 Popular Ai Assistants." *Built In*, 2023, builtin.com/artificial-intelligence/ai-assistant.
3. "Unlock Your Productivity Potential with Slack Platform." *Slack API*, Sept. 2023, api.slack.com/.
4. Ragala, Bala Krishna. "Software Engineer Challenges and Solutions to Overcome." *KnowledgeHut*, 4 Sept. 2023, www.knowledgehut.com/blog/web-development/software-engineer-challenges.
5. Lucia, Andrea De, and Abdallah Qusef. "Requirements Engineering in Agile Software Development." Journal of Emerging Technologies in Web Intelligence, vol. 2, no. 3, 20 Aug. 2010, https://doi.org/10.4304/jetwi.2.3.212-220.