TaskFlow

A TODO List Messaging Application

Andrew Tang
Computer Science
Virginia Tech
Blacksburg, VA USA
at346@vt.edu

Daniel Abaye
Computer Science
Virginia Tech
Blacksburg, VA USA
danielabaye@vt.edu

Mazin Abdelrahman
Computer Science
Virginia Tech
Blacksburg, VA, USA
mazintarawa@vt.edu

Dylan Lau
Computer Science
Virginia Tech
Blacksburg, VA USA
dylanlau@vt.edu

ABSTRACT

Software engineers can waste a bunch of time sending emails to tech leads and project managers about project specifics or what they need to do. TaskFlow is a TODO list messaging application that will make communication and task tracking easier and more efficient.

INTRODUCTION

Software engineering teams can end up wasting a lot of time sending emails about clarifying tasks. This could be asking questions about specific project requirements, what task should be worked on next, and more. With TaskFlow, tech leads, and project managers will be able to send out interactive TODO lists. This will save time on the need to send a bunch of emails.

The lists will be interactive. Project managers (PM) will be able to send out customized individual TODO lists to team members. Developers will be able to check off a task and a PM will be able to see its status in real time. Developers can comment on TODO lists if they have any inquiries. PMs will be able to see these messages and reply through the app or make any necessary updates to the list. If the question is more urgent or requires immediate feedback, integration with Zoom or Microsoft Teams will be available. Tasks can be hierarchically organized to improve workflow. PMs can view active and completed TODO lists in a visual dashboard that will improve task tracking and organization. These TODO lists will improve the workflow of developers by allowing them to focus less on emails and more on their work and what they should complete

Ultimately TaskFlow is a specialized method of communication for software developers. It will replace long chains of emails that become hard to read with a more clear and concise task messaging app.

EXAMPLE

During a sprint meeting a team of developers plans the next tasks that they need to work on. The developers take notes, but sometimes they interpret the tasks incorrectly or don't write the tasks down in enough detail. With TaskFlow the project manager or tech lead can send out personalized TODO lists to each developer. Now the developers can ensure that they have the correct tasks written down. A developer wants to make sure they understand their task correctly or have a question? They can make comments directly on the task, which the sender of the list can see. In response the sender can send a message back or schedule a meeting through the application to work out any concerns.

RELATED WORK

Slack is a popular messaging platform that has gained traction in software development teams due to its less formal and time-efficient nature compared to traditional email [1]. By allowing for real-time communication through channels and direct messages, Slack reduces the back-and-forth of emails, enabling team members to seek clarifications or provide updates more swiftly. This aligns with the goals of *TaskFlow*, which aims to further streamline communication by integrating task management directly into the messaging framework. While Slack facilitates communication, it lacks structured task management capabilities, which can lead to important tasks being overlooked or lost in chat threads. *TaskFlow* addresses this by providing an interactive TODO list feature, ensuring that tasks are clearly outlined and tracked alongside communication.

Jira, on the other hand, is a robust project management tool specifically designed for tracking and recording the progress of work in software development [2]. It allows teams to create detailed task lists, assign tasks to team members, and monitor the status of ongoing work. However, its complexity can sometimes hinder quick communication between team members. While it excels in task tracking, Jira does not prioritize immediate interaction or address the real-time inquiries that often arise during development.

IMPLEMENTATION

TaskFlow follows a Client-Server Architecture. The client-side, built with React and JavaScript, renders the user interface and handles user interactions, such as creating tickets or sending messages. Communication with the server is managed through RESTful APIs to perform operations like fetching ticket data, updating statuses, and handling messages. On the server side, the application consists of several key components: the Ticket Management Service to manage ticket creation and updates, the Messaging Service to enable real-time communication using WebSockets, and the User Management Service to handle authentication, authorization, and user roles. A relational database, such as PostgreSQL, stores ticket data, while a NoSQL database like MongoDB is used for chat messages to ensure scalability and efficient access. Push notifications are implemented for real-time alerts, and modular server components interact to ensure clear separation of concerns and maintainability.

We followed an Agile development process, with work organized into bi-weekly sprints. Each sprint begins with planning to outline what will be accomplished during the meeting. Next a review of the previous meeting is done to analyze what tasks were completed. During velocity tracking we estimated the time frame the next set of tasks would take to finish. We also delegated tasks during this time. Capacity planning estimated each of our availabilities for the current sprint. Finally, we finished the meeting by identifying potential risks that could be encountered during the current sprint. Next a set of steps to mitigate those risks were planned out. Ultimately sprint meetings helped us plan and organize. Since we all had other commitments it was important to meet and express our availability to delegate tasks based on that. Reviewing the previous sprint allowed us to make improvements, which increased productivity.

Testing was approached using black-box testing to ensure that we validated the functionality from a user's perspective. We outlined our test cases to reflect the core tasks and user interactions within the app, focusing on ensuring that TaskFlow performs as expected from the standpoint of stakeholders. Our black-box test plans were intentionally designed using plain language to make sure the test cases were clear to all stakeholders, regardless of their technical background. Specifically for our test cases we covered key workflows such as task prioritization, list creation, messaging, and task completion, ensuring each feature aligned with user expectations and functional requirements.

DEPLOYMENT AND MAINTENCE

The tech stack for deployment will use GitHub for version control and GitHub Actions for CI/CD. Using GitHub Actions will allow our team to push updates and fixes to TaskFlow at a much faster rate and minimize downtime. GitHub has the benefits of automated testing, which will prevent new updates or fixes from breaking other parts of the program. Due to our application being an integral part of our user's software engineering process, blue-green deployment will be used. We would not want a new version to cause something catastrophic to occur then disrupt our user's workflow by preventing them to see their TODO lists. Blue-green deployment will allow us to see if a new version in staging is working as intended. Once functionality is verified it can be pushed to production.

The steps below are what the deployment for our application would look like [3]:

1. Code commit to repository that triggers GitHub Actions
2. GitHub Actions runs automated test on the changes
3. Code is deployed to green cluster for live testing
4. Using various tools, such as Azure Load Testing, to verify performance and functionality between the blue and green cluster
5. Traffic is switched to green cluster
6. Rollback if there are issues with the green cluster (optional)
7. Retire blue cluster

We will use perfective software maintenance to optimize performance, scalability, and user experience in TaskFlow. By leveraging GitHub Actions for CI/CD and blue-green deployments, we can safely test and introduce improvements without disrupting users. Tools like Azure Load Testing will help identify areas for enhancement, ensuring efficient resource use and faster deployment times. This approach allows us to continuously refine the system, adapt to user needs, and maintain TaskFlow as a reliable, high-performing tool in their workflow.

CONCLUSION

TaskFlow solves the inefficiencies of traditional task communication by integrating interactive TODO lists with real-time messaging, reducing reliance on long email chains and improving workflow clarity. Implemented using a modern Client-Server Architecture with technologies like React, RESTful APIs, and WebSockets, TaskFlow enables seamless task tracking, updates, and inquiries. Following an Agile development process and validated through black-box testing, the project ensures reliability and alignment with user needs. With a robust CI/CD pipeline using GitHub Actions and blue-green deployments, TaskFlow minimizes downtime and allows for safe updates. This solution streamlines task management and communication, enhancing productivity for software engineering teams.

REFERENCES

[1Slack, "What is Slack?," *Slack Help Center*, 2022. https://slack.com/help/articles/115004071768-What-is-Slack-

[2] Atlassian. All great projects. *Atlassian*. https://www.atlassian.com/software/jira.

[3]vimorra, "Blue-green deployment of AKS clusters - Azure Architecture Center," *learn.microsoft.com*. https://learn.microsoft.com/en-us/azure/architecture/guide/aks/blue-green-deployment-for-aks