TagTeam Dashboard

Final Report

Authors:

Jacob Morroni, Rishi Singh, Nebiyu Abreha

Abstract:

The TagTeam Dashboard solves critical collaboration problems for software development teams: uneven workloads, ownership ambiguity, and scattered help requests. It unifies all these in one place by letting developers tag problem code sections, seamlessly integrate GitHub to reflect changes in real-time, and provide a reward system to encourage peer collaboration. Feedback from previous milestones highlighted that there should be more usability and integration with the workflows being used. These insights have now been integrated to improve the capabilities of the dashboard and offer a better user experience. The following report outlines the rationale for the project, design issues, implementation processes, deployment strategies, potential opportunities regarding future enhancements, and addressing limitations found during development.

Introduction:

Poor task management, communication breakdowns, and lack of clear ownership in client-based software development have led teams to experience crippling delays and inefficiencies. Developers struggling with issues in their code can't find timely help that can stall progress and delay project delivery. Traditional tools such as JIRA and GitHub Issues offer task management and issue tracking but fall short in enabling direct code-specific collaboration. The TagTeam Dashboard fills these gaps by allowing developers to tag sections of code where they need help, integrates with GitHub to keep tasks in sync, and incentivizes developers to help their peers.

This is an Agile project, and as such, it has been iterating through multiple development sprints, with continuous user feedback. The core functionalities of the dashboard-tagging, GitHub integration, and reward system-are put together with the intent to ease collaboration and make development workflows smoother. The TagTeam Dashboard solves these common pain points by increasing productivity, enhancing teamwork, and improving project outcomes.

Working in a development team on a complex project with a tight deadline, one of the developers encounters a bug that persists in a critical section of code during coding. Since there is no clear way to ask for help, he either wastes a lot of time trying to troubleshoot by himself or sends a vague request in some general communication channel. This lack of specificity leads to delays, miscommunication, and frustration between team members. TagTeam Dashboard solves this by allowing the developer to tag the exact section of the code where the problem arises, describe the issue in greater detail, and immediately notify your team. Other developers can then respond as soon as possible, thereby offering solutions and collaborating efficiently. The reward system is native, and team members like to help each other out for building a sharing and helping culture. This saves hours conventionally spent in looking for information, improves communication, and resolves issues timeously to keep the project on track.

Related Work:

There are various tools that implement individual parts of our program but nothing that our team has seen that combines them all in a standalone product. Nevertheless, because our program combines a code editor, discussion board, and version control system, it can be compared to other services such as GitHub, Slack, and Visual Studio Code.

GitHub

Our team chose to integrate GitHub into the product because it is the leading Git version control system and it is straightforward to use. But, it doesn’t match the type of dashboard our team is implementing with our strategy for organization and accessibility. Therefore, while

GitHub has many similar features and abilities, we wanted to more easily mesh the discussion board and editor into the GitHub suite.

Slack

Slack is used by many development teams for communication and collaboration across many industries and company sizes. The issue is that it is difficult to discuss software related issues without posting screenshots and images which can clutter the text stream. Therefore, while Slack is a strong tool, it doesn't allow for the level of technicality that the TagTeam Dashboard strives to provide.

Visual Studio Code

Visual Studio Code is known as one of, if not the best code editor on the market, due to its simplicity, yet power through extensions and developer tools. Its only weakness from the team's perspective is its lack of collaborative options. This is why the TagTeam Dashboard was implemented to expand on its features to increase a team's productivity.

Design:

High-Level Design

The team used the Model-View-Controller architectural pattern to structure the system. The reason for this is because this pattern separates the concept into 3 main sections. This choice facilitates scalability and maintainability, key for evolving collaborative tools. It allows for pinpointing of errors or bugs, and is not needed to refactor the entire product if major issues arise. We believe this architecture would be the strongest option for our design.

Implementation Process

The team would pursue a behavior-driven development process in order to develop with both tests and features in mind. Since the team performed requirements engineering, we started with a considerable amount of behavioral specifications rather than tests and code. Therefore, this implementation process would be beneficial, because the product would build off of these requirements rather than a code base that was not accessible. Additionally, this method would allow the team to target specific user requests by laying out all of the intended features of our product and using that as the foundation for success.

Testing Approach

The team used a black box test plan to lay out key areas of testing to ensure proper functionality of the product in many scenarios. A black box plan was specifically used to highlight inputs and outputs of the program without seeing the actual inner workings of the software. If tests failed, the team would investigate the software, but from an overarching perspective, the internal visibility is unnecessary. Tests were categorized by types such as user interface, integration, communication, overload, security, etc. in order to organize the tests into packages for specific teams.

Deployment/Maintenance:

Deploying and monitoring our dashboard product will be guided by DevOps principles and CI/CD practices. Continuous Integration (CI) automates building, testing, and analyzing every code change, ensuring early defect detection and enabling low-risk releases. Continuous Delivery keeps the software in a deployable state, making quick and low-risk updates possible. Deployment strategies like blue-green deployment and canary deployment help minimize downtime and user impact by testing changes in staging environments and rolling out updates incrementally before full production. Infrastructure as Code (IaC) ensures consistent and repeatable deployments across different environments. Performance tracking systems and automated testing strategies, such as fuzz testing and test case prioritization, further enhance system stability. By automating as much as possible and fostering strong collaboration between development and operations teams, we can ensure predictable, reliable deployments while continuously improving the user experience.

Conclusion:

Some limitations faced by the product include unsupported file types when uploading code for a repository and an inability to precisely customize the dashboard. In the future, the team would implement functionality to support any file type such as images or video with optical character recognition to convert these file types into code-based files. Additionally, alternative components could be developed such as a statistics panel for project managers or a test performance panel for maintainers. These panels could be toggled on or off depending on the user's roles to further specialize the dashboard for each member of a team. In this way, our product could be more widely used and solve many more problems in software development industries.

In conclusion, the TagTeam dashboard solves several common problems faced by developers such as uneven workloads, overflowing help requests, lack of support between teams, and more. The dashboard incentivizes developers to support each other in order to prevent falling behind while also giving them a reason to do so. Encouraging collaboration between developers would lead to a more productive work environment, allowing development to be more efficient and collaborative. Integration of commonly used tools such as GitHub would only further assist this process, making the collaboration even more seamless.

References:

Blischak, J. D., Davenport, E. R., & Wilson, G. (2016, January 19). *A quick introduction to version control with Git and github*. PLoS computational biology. https://pmc.ncbi.nlm.nih.gov/articles/PMC4718703/

Johnson, H. A. (2018, January). *Slack*. ResearchGate. https://www.researchgate.net/publication/322451553_Slack

Tan, J., Chen, Y., & Jiao, S. (2023, March 10). *Visual studio code in computer science*. arxiv. https://arxiv.org/pdf/2303.10174