TagTeam Dashboard
Improving Team Collaboration

Authors:
Ali Manzoor, Jacob Morroni, Rishi Singh, Nebiyu Abreha

Process Deliverable I:

We chose to implement an agile SE process for developing TagTeam Dashboard. For the process deliverable, we are submitting a retrospective analysis and list of prioritized tasks for the next sprint planning milestone.

Retrospective Summary:

Achievements:

- The TagTeam Dashboard successfully addresses critical collaboration challenges such as uneven workloads, scattered help requests, and lack of effective communication in client-based development. The team has effectively conceptualized an idea for a solution to improve collaboration through tagging troublesome code sections, integrating GitHub for ease of updates, and rewarding peer assistance.
- Tools Evaluation: A comparative analysis of existing tools like JIRA, GitHub Issues, and Slack showed their limitations in promoting direct code-related collaboration. TagTeam bridges this gap by enabling tagging specific code regions for help and fostering a reward system for collaboration.

Challenges:

- Workload Distribution: Pre-deployment phases often resulted in uneven workloads due to vague task ownership. This issue hindered the smooth distribution of tasks.
- Help Requests: Scattered and unstructured help requests caused delays and inefficiencies.
- Tool Integration: Existing tools lacked full integration capabilities, especially in task management, direct collaboration, and GitHub linkage.

Prioritized Tasks for the Next Sprint Planning Milestone

1. Develop Tagging Feature
    a. Allow developers to tag problematic sections of code for collaboration. This feature should be easy to use and directly linked to GitHub.
    b. Priority: High

2. GitHub Integration
    a. Implement integration with GitHub which enables team members to edit code directly from the dashboard and keep track of tasks without switching platforms.
    b. Priority: High
3. Reward System
    a. Develop and implement a feature to reward developers for helping their peers, promoting a collaborative environment.
    b. Priority: Medium
4. Wireframes/Mockup Design for Dashboard UI
    a. Create wireframes and mockups to visualize the dashboard layout, tagging process, and task management flow.
    b. Priority: Medium

Requirements Analysis:

1. Hypothetical Non-Functional Requirements

Performance Requirement:

The system shall not exhibit response time degradation under the condition of 500 users at a time.

Security Requirement:

The system should also offer two-factor authentication to all the users logging in to the dashboard for better security.

Usability Requirement:

85% of the new users should be able to successfully execute their first tagging task within 5 minutes after login.

Scalability Requirement:

The system shall be designed such that it can scale to accommodate 50 concurrent projects, which will have different numbers of sub-teams and different loads of work in the different teams.

Maintainability Requirement:

The code base shall be modular so that bugs can be isolated and fixed within 24 hours.

2. Hypothetical Functional Requirements

The system shall allow developers to tag relevant portions of code, requesting help from other members.

The system shall notify all the developers who are working on a project when a new tag for code is added.

The system shall be integrated with GitHub so that users can directly edit the code base from the dashboard.

System shall keep the count of issues tagged which are dealt by developers and award points part of the reward system.

System shall allow project managers to assign tasks to sub-teams; the progress of the task can be viewed in real time 3. Use Cases

3. Use Cases

Use Case 1: Tagging a Section of Code

Preconditions:

The developer is authenticated and has access to the code.

Main Success Scenario:

The developer chooses the problematic section of code

The developer clicks the "Tag for Help" button.

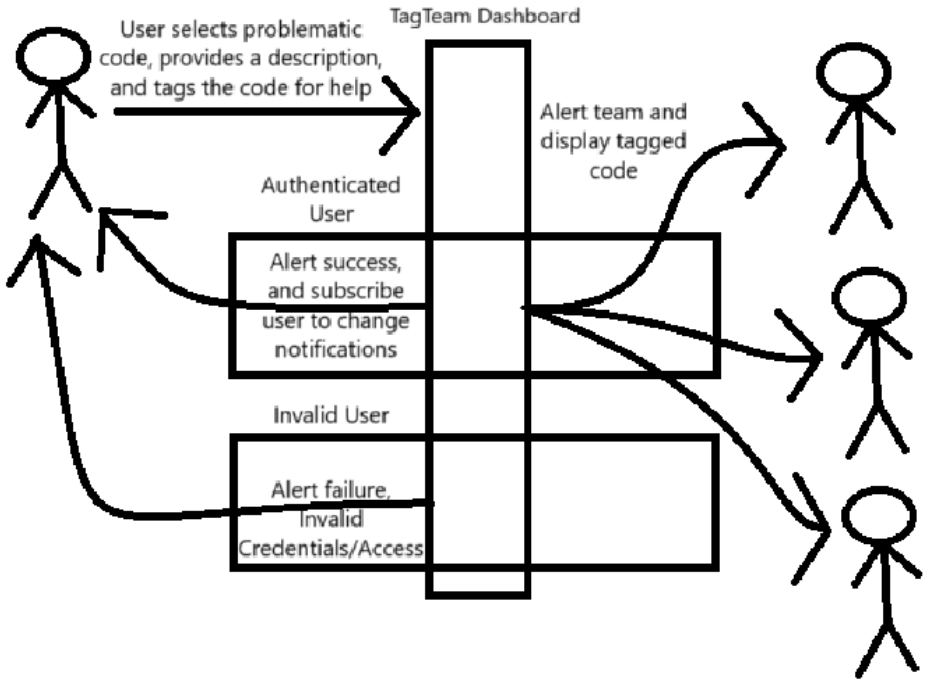
The system asks the developer to provide a description of problem s/he is facing

The developer submits the tag

The system alerts the rest of the team.

Post-conditions:

The code is tagged and team members are notified



Use Case 2: A developer replies to a tagged piece of code.

Prerequisites:

The developer has access to the project in which he found the tagged piece of code.

Main Success Scenario:

The developer clicks on "Help Needed".

The developer chooses one tag to view.

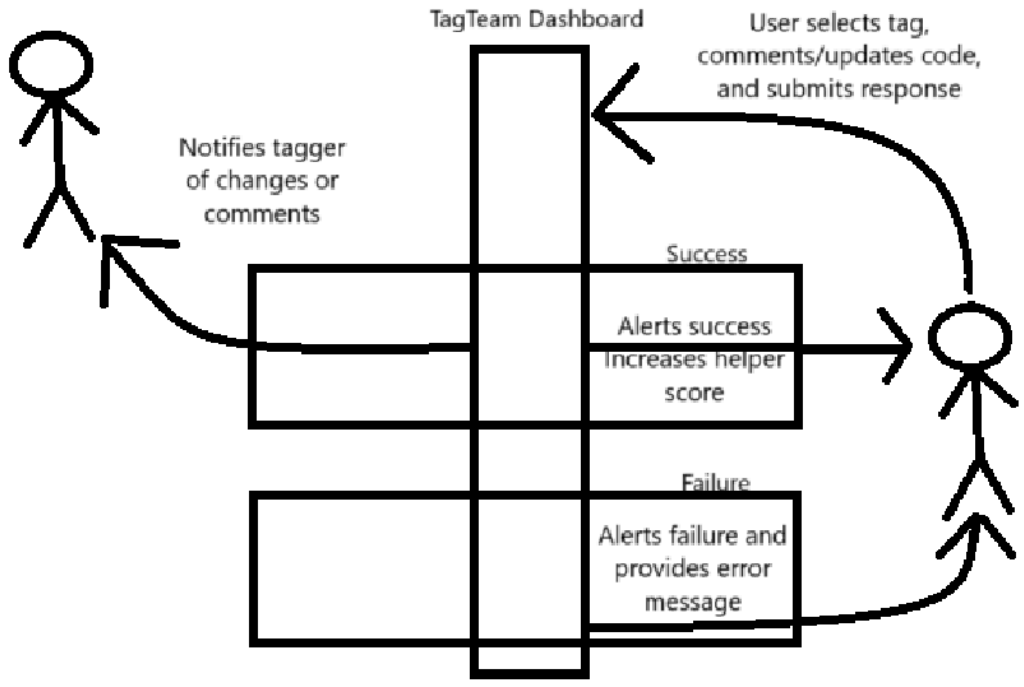
The developer reviews the code, adds comments, or updates

The developer submits his response

The author of the original code is notified

Postconditions:

The problem that led to the tagging of the code is solved and helper is rewarded



Use Case 3: A developer fixes a piece of code and pushes it to Git

Prerequisites:

The developer has access to the project in which he found the tagged piece of code and has access to the repository.

Main Success Scenario:

The developer reviews the code and makes sure its ready to add

The developer saves the code.

The developer adds the code

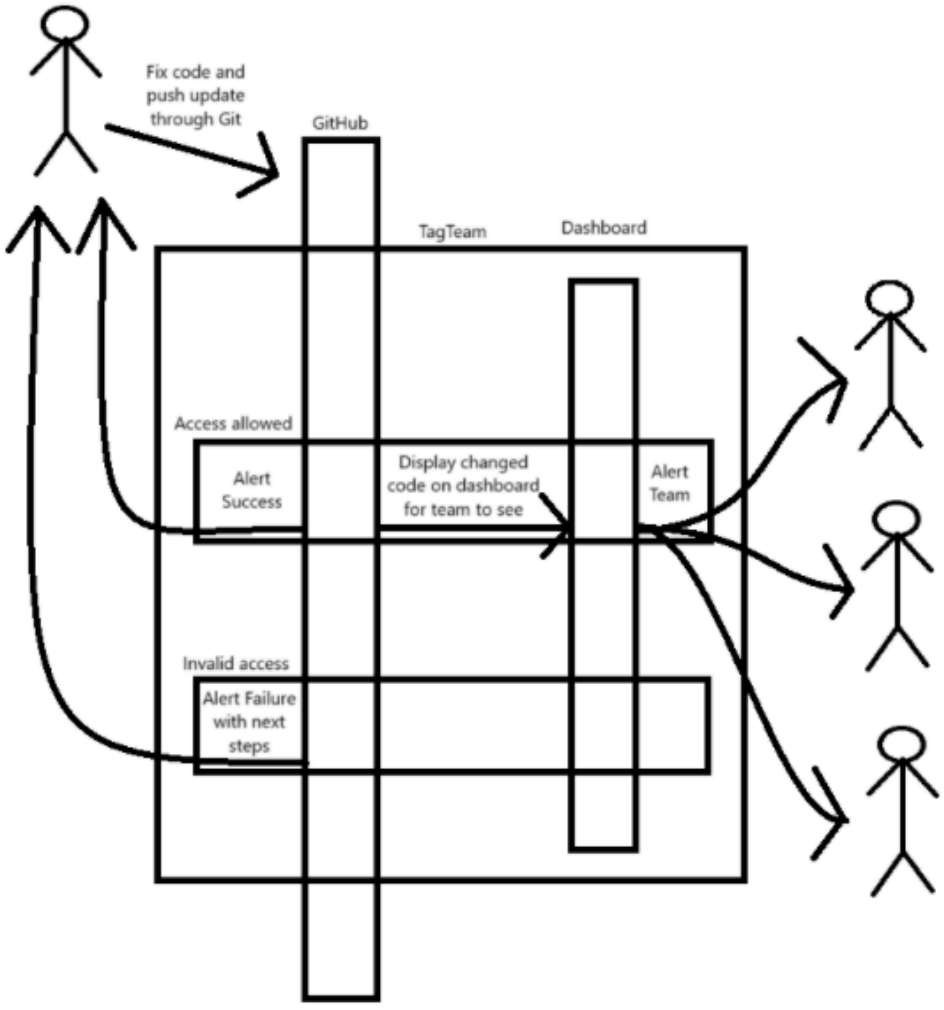
The developer pushes the code

The developer ensures the correct code is on the repository and can be accessed by others

The contributors are notified.

Postconditions:

The code is fixed and is updated on the repository.



Use Case 4: Program manager observes project progress and collaboration.

Prerequisites:

The program manager can see how many developers have collaborated using the dashboard.

Main Success Scenario:

The program manager reviews how many developers have used the dashboard

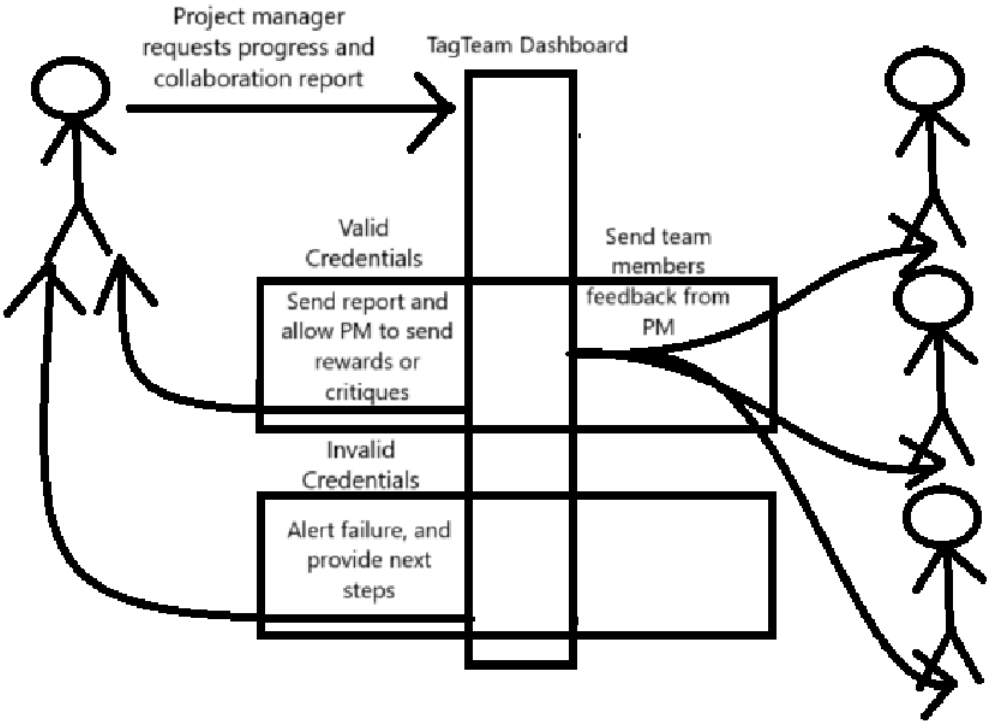
The program manager reviews the code changes made

The program manager determines whether the collaboration is sufficient enough for a reward

The program manager distributes rewards to developers helping others

Postconditions:

The errors are handled and rewards are distributed to developers.



Use Case 5: Developers discuss errors using a chat for tagged code

Prerequisites:

Developers have access to the code and the chat made for tagged code.

Main Success Scenario:

The developer clicks the “Chat” button to open the chat

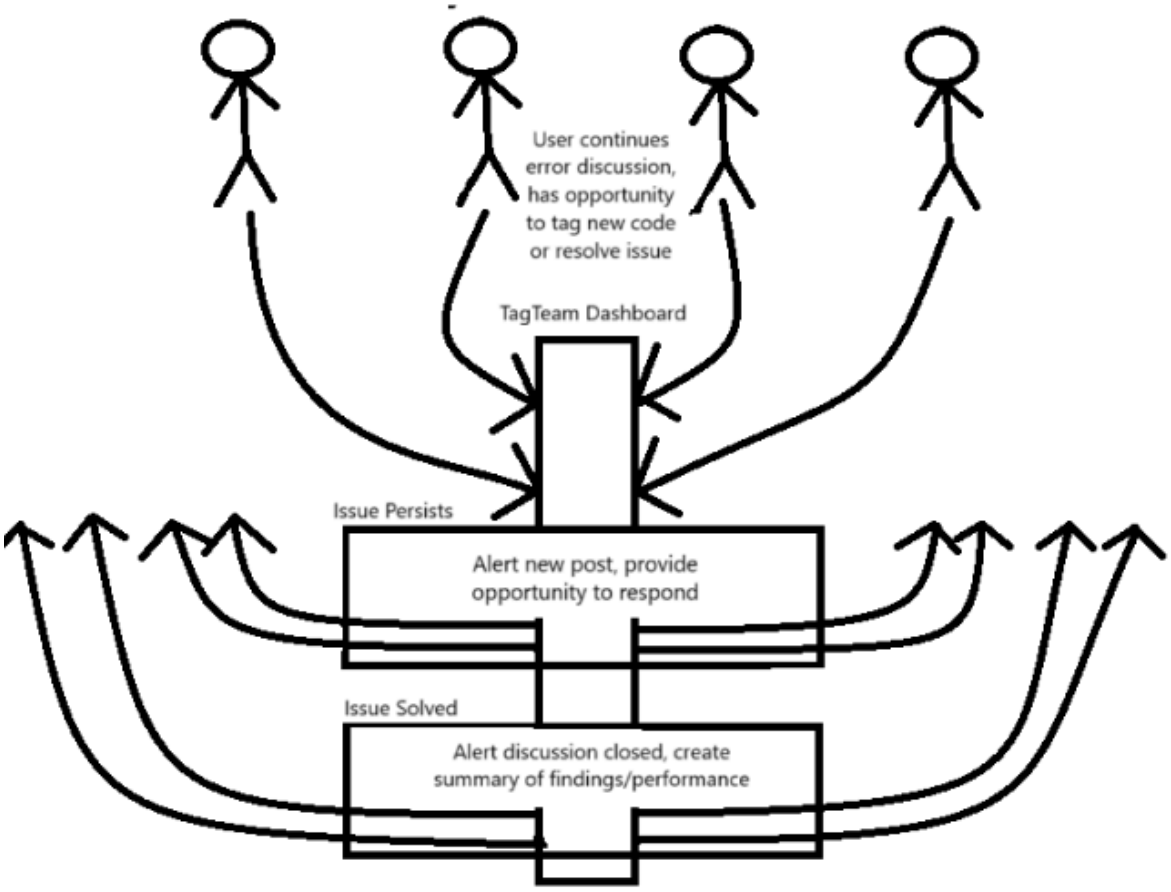
The developer sends messages to the chat to communicate about how to fix errors

The developers sends a code snippet which is integrated with the chat to reference pieces of broken code

The developers use this collaboration to fix the code and then closes the ticket

Postconditions:

The errors in the code are better communicated, and the solutions are more diverse.



Requirements Specification:

User Stories

User Story 1:

As a Developer, I want to label portions of code that I am experiencing difficulties in so that I can promptly get assistance from my peers.

Acceptance Criteria: Once I assign a label to a piece of code, then the team gets that and the code is pushed on the "Help Needed" list.

User Story 2:

As a Project Manager, I need to assign tasks to subteams so that I can ensure to allocate the workload efficiently.

Acceptance Criteria: The tasks assigned will be available in the task board of the subteam and they get status changes in real time

User Story 3:

As a Developer, I'd like to enable resolution of other tagged issues so that the issues begin to get resolved.

Acceptance Criteria: The original developer should be notified upon resolution of a tagged issue. Contribution logged

User Story 4:

As a Project Manager, I want GitHub to be integrated directly into the dashboard so that developers don't have to continuously switch between their platforms in order to manage their code within it.

Acceptance Criteria: Every time changes push into GitHub, the status of the Task automatically gets updated along with the Code change.

Function Point Estimation

User story	inputs	outputs	Inquiries	Files	Interfaces	Function Points
Tag Code Section	1	1	1	0	0	3
Assign Tasks to Sub-teams	1	1	0	1	1	4
Assist Other Developers	1	1	0	0	0	2
GitHub Integration	2	1	1	1	2	7

In the project TagTeam Dashboard, a function point profile was built for each user story based on five functional components: inputs, outputs, inquiries, files, and interfaces. These components quantify the complexity and effort required to develop each feature; the higher the function points, the more resource-intensive the task will be.

For the "Tag Code Section" feature, estimated at 3 function points, the complexity is moderate. Here, developers will input tagged code sections with requests for assistance, the system will output notifications to the team, and the developers may view-or inquiry-the tagged section to understand the issue. No large file storage or interfaces are required, hence keeping the overall complexity lower.

Task assignment to sub-teams: Estimated at 4 function points, this feature needs more effort because the assignments of tasks are complex and need to be shown on the task board of the sub-team. The procedure consists of inputs from project managers in creating tasks, the assigned tasks displayed as output, and file storage to maintain the assignment of tasks. Added to that is an interface for managers and sub-teams to interact with tasks, further increasing the function point total count.

The "Assist Other Developers" functionality, with an estimate of 2 function points, will also be quite simple. A developer will provide comments or solutions to the tagged issues. Outputs will involve notices for the original tagger. The file storage will be minimal, and the interface requirements will likewise be low; most of the interactions will revolve around the tagged issues.

The most complicated feature is "GitHub Integration," given that it has 7 function points. It needs not only developer inputs but also GitHub inputs to synchronize changes and outputs to present the task status in real time. In addition, users can query updated task statuses, for which storage is needed to manage the various versions. GitHub integration also requires multiple interfaces to deal with syncing and code, hence being the most resource-intensive feature.