Progress Point: Real-Time Communication Platform

Mihir Palyekar
Computer Science
Virginia Tech.
Blacksburg,USA
mihirlahu@vt.edu

Parth Bapat
Computer Science
Virginia Tech.
Blacksburg,USA
bapatparth@vt.edu

Vivek Joshi
Computer Science
Virginia Tech.
Blacksburg,USA
vivekj@vt.edu

Sushma Deegoju
Computer Science
Virginia Tech.
Blacksburg,USA
sushmad@vt.edu

***Abstract*—In the current fast-paced software development environment, it is critical for team members to stay aligned and up-to-date on each other's progress. Sending emails to everyone and frequent status meetings, may not be feasible and fall short in facilitating real-time collaboration and communication. This can hamper the performance of engineering team, leading to missed deadlines and quality issues. The proposed solution is the development of a real-time collaboration and communication platform called Progress Point for software engineers. The platform will enable software engineers to post updates about their tasks, subscribe to updates from other team members, and collaborate on problem-solving in real-time. In addition, the platform will provide a central location for sharing information and files, reducing the time and effort required to find and share relevant information. With Progress Point, we aim to improve teamwork and productivity in software development projects. This platform will help software engineers stay aligned, work together to solve problems efficiently, and deliver high-quality software products on time.**

I. Introduction

Software development is a complex and collaborative process that involves multiple teams and stakeholders. With world moving towards remote working and virtual environments, streamlined communication becomes cardinal for any team to deliver results. Virtual meetings are a good way to communicate but it is not feasible to have meetings with all team members multiple times in a day to discuss task-related information. Software engineers may encounter challenges that they wish to discuss, or they may wish to share task-related updates with stakeholders. To promote teamwork and collaboration, a platform with real-time communication capabilities becomes a necessity. [1] To address the challenges related to communication, we propose The Progress Point. It is real-time collaboration and communication platform for software engineers. This platform will provide a central location for team members to communicate, collaborate, and share information. The platform will enable software engineers to post real-time updates about their tasks, subscribe to updates from other team members, and collaborate on problem-solving. The proposed platform will solve several problems faced by software engineering teams. One of the biggest challenges is the lack of real-time communication and collaboration, which frequently results in missed deadlines and poor quality. With the Progress Point, team members can stay up-to-date on each other's progress and work together to solve problems, leading to improved productivity [2] and quality [3]. This can also help in inter-dependent tasks where team members can immediately start the next task in the pipeline after the previous task is finished. Another challenge faced by software engineering teams is the lack of a central location for sharing information and files. Without a central location, team members waste time searching for information and files, leading to lost productivity and frustration. The proposed platform will provide a central location for sharing information and files, reducing the time and effort required to find and share relevant information. Finally, the proposed platform will help to remove blockers and improve collaboration across the organization. In many software development projects, multiple teams and departments are involved, including engineers, designers, product managers, and QA teams. The Progress Point will provide a platform for cross-functional collaboration, allowing team members from different departments to communicate, share information, and stay aligned on project goals [4].

II. Related Work

[1] The research paper presents a case study on the impact of regularly updating and tracking tasks in agile software development. The authors found that regularly updating and tracking tasks can improve team collaboration, reduce rework, and increase software quality by ensuring that team members have accurate and up-to-date information about project status and task progress.

[3] The research paper examines the relationship between collaboration and software development performance based on survey data collected from software development teams. The authors found that collaboration has a positive impact on performance, leading to increased productivity, better quality, and faster delivery times. Collaboration improved communication and knowledge sharing among team members, resulting in more efficient work processes and better decision-making.

[2] This paper talks about the importance of communication between teammates while creating a software development team.It is noted that the test subjects only noticed a gradual increase in productivity while practicing the appropriate communication methods mentioned in the paper, as compared to a immediate increase in productivity.

[4] This research paper focuses on how cross-functional agile teams communicate and update their task progress in software development projects. The present paper aims at analyzing how large organizations, characterized by distributed

and cross-functional teams, can cultivate an agile environment and better communication methodologies where inter-individual knowledge exchanges are encouraged.

III. High Level Architecture Design

Building an web application like progress point warrants the use of an MVC like architecture. Here are a few components that we plan on building to make the final software, and how they fit together:

- Front-end: The front-end of the application is the part that users interact with directly. It is responsible for displaying the UI components, managing user input and displaying the results of the back-end operations. We will be using handlebars with server side rendering for this project.
- Back-end: The back-end is responsible for processing user requests and generating responses. It consists of servers, APIs, and databases. We plan on using Node.js with an express framework to give support for server side rendering.
- Authentication: The authentication system is responsible for verifying the identity of users and granting access to authorized resources. It can be implemented using technologies such as OAuth 2.0, JSON Web Tokens (JWT), or OpenID Connect. For this project we plan on implementing a JWT token to authenticate the API and user activities.
- Data Storage: The data storage system consists of a database where user-generated data is stored. This can include user profiles, posts, updates, likes, and reposts. We will be using MongoDB for storage, as it is a NoSQL database, which is appropriate for our project.
- Caching: The caching system is responsible for improving the performance of the application by storing frequently accessed data in memory. This can include data such as user profiles, Posts, and trending topics. We are planning to implement caching using Redis for the user profile.

These components work together to create a seamless user experience on the front-end while ensuring scalability, reliability, and security on the back-end. The specific technologies and architecture we use may change depending on the specific requirements and constraints of the project.

IV. Design constraint and guidelines

While building Progress Point, we have considered the following constraints and guidelines:

- Separation of Concerns: We plan to use MVC architecture style as explained to ensure separation of concern. To maintain this separation, it's important to ensure that each component only contains code relevant to its specific responsibilities.
- Modularity: To build a scalable and maintainable application, it's important to break down the system into smaller, modular components. This allows different parts of the application to be developed and tested independently, and makes it easier to make changes without affecting the rest of the system.
- Scalability: To ensure that the application can handle a large number of users and a high volume of data, it's important to design the application with scalability in mind. This may include techniques like sharding the database, using caching, and using load balancing to distribute traffic across multiple servers.
- Security: Building a platform like Progress Point requires a strong emphasis on security to protect user data and prevent unauthorized access. This may involve implementing measures such as encryption, authentication, and access control.
- Testing: To ensure that the application is functioning as expected, it's important to establish a comprehensive testing framework. This may include unit tests, integration tests, and end-to-end tests, and should cover all components of the system.

V. Relevant design patterns

For the progress Point, we have considered following design patterns that can be used for development. A creational pattern can be used to manage the creation of new messages and users. A behavioral pattern can be leveraged to streamline communication between objects and real-time notifications.

- Creational Pattern: Creational design patterns provide ways to create objects in a flexible and reusable manner. Creational patterns like the factory pattern can be useful for creating objects that are similar in structure but differ in behavior, such as text messages and images. The singleton pattern can be used to ensure that only one instance of a particular object is created throughout the application. This could be useful for managing global objects such as the user database, message queue, or notification system.
- Behavioral Pattern: application would involve a lot of communication between different objects, such as users, messages, and notifications. To manage this communication effectively, one could use behavioral patterns such as the observer pattern to implement real-time updates and notifications, or the mediator pattern to handle the exchange of messages between different objects in a centralized manner.

References

[1] F. Kortum, J. Klünder, and K. Schneider, "Behavior-driven dynamics in agile development: The effect of fast feedback on teams," in *2019 IEEE/ACM International Conference on Software and System Processes (ICSSP)*, 2019, pp. 34–43.

[2] J. M. Hogan and R. Thomas, "Developing the software engineering team," in *Proceedings of the 7th Australasian conference on Computing education-Volume 42*, 2005, pp. 203–210.

[3] J. D. Herbsleb and A. Mockus, "An empirical study of speed and communication in globally distributed software development," *IEEE Transactions on software engineering*, vol. 29, no. 6, pp. 481–494, 2003.

[4] C. Khalil, V. Fernandez, and T. Houy, "Can agile collaboration practices enhance knowledge creation between cross-functional teams?" in *Digital Enterprise Design and Management 2013*, P.-J. Benghozi, D. Krob, and F. Rowe, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 123–133.