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 today's rapidly evolving software development landscape, ensuring that team members are well-informed and in sync on their respective progress is critical to achieving project objectives. Traditional methods such as sending emails or hosting frequent status meetings have been found lacking in facilitating real-time collaboration and communication, resulting in sub-optimal team performance, missed deadlines, and quality issues. The proposed solution is "Progress Point", which is a centralized platform for teams to communicate and collaborate on software development projects in real-time. The platform will enable software engineers to be a part of multiple team rooms, post updates about their tasks, and collaborate on problem-solving in real-time. The platform will enable developers to react and comment on each others' posts and communicate. 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. To address the challenges related to communication, we propose The Progress Point.

Progress Point 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 [1] and quality [2]. It allows members to react and comment on posts allowing users to get quick feedback and facilitate faster communication. Developers can also be part of multiple chat rooms to get updates from different teams which can help in cross-functional communication. 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. 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 updates, 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 [3].

II. Motivating Examples

- Real-time communication: Real time communication is essential for software development teams to be effective in the fast-paced world of today. The Progress Point gives team members a platform for real-time communication and collaboration, enabling them to keep informed about each other's progress and co-operate to find solutions to issues.
- Cross-functional collaboration: Several teams and departments, such as engineers, designers, product managers, and QA teams, are involved in many software development projects. The platform offered by the Progress Point allows team members from different departments to communicate, share information, and keep their focus on the project's goals.
- Facilitating team building: Progress Point can also be used to facilitate team building and socialization. For example, team members can create channels dedicated to hobbies or interests outside of work, helping to build a sense of community within the team.

- Increased productivity and quality: The Progress Point can assist increase productivity and quality in software development projects by offering a platform for in-the-moment communication, centralized information sharing, and cross-functional collaboration. The platform can support teams in completing projects on time and at a higher standard by allowing team members to collaborate more successfully.
- Communicating with remote teams: Many software engineering teams work remotely, and Progress Point platform can help to bridge the gap between team members who are physically located in different locations. With features like video and voice calls, remote teams can stay connected and collaborate effectively.
- Onboarding new team members: When new software engineers join a team, it can be challenging to get up to speed quickly. Progress Point can be used to share information and resources with new team members, making it easier for them to get up to speed and start contributing to the team.
- Enhanced project transparency: The Progress Point can improve project transparency by giving team members and stakeholders access to information on the status, deadlines, and deliverables of the project. This can promote trust between team members and stakeholders and guarantee that everyone is clear on the objectives and expectations of the project.
- Sharing feedback and recognition: Progress Point can be used to share feedback and recognition with team members, helping to build a culture of continuous improvement and positive reinforcement. This can help to boost morale and improve team dynamics.

III. Background

Effective communication and collaboration are critical to the success of software engineering projects. Several studies have highlighted the importance of communication in software development. Communication breakdowns were a major cause of project delays and budget overruns [4]. And better communication and collaboration led to improved software quality [5].

Real-time communication and collaboration platforms have become increasingly popular in software engineering. These platforms provide a centralized location for team members to communicate, collaborate, and share information. Several studies have investigated the use of these platforms in software development. Real-time collaboration tools improved communication and collaboration among distributed teams [6] improving team performance and reduced project delays [7].

Other studies have investigated the use of specific real-time collaboration platforms in software engineering. For example, the use of Slack in software development and found that it improved communication and collaboration among team members [8] . Likewise, [9] investigated the use of Microsoft Teams in software development and found that it improved team productivity and communication.

In summary, research has shown that effective communication and collaboration are critical to the success of software engineering projects. Real-time communication and collaboration platforms can help improve communication, collaboration, and productivity in software development. Specific platforms, such as Slack and Microsoft Teams, have been shown to be effective in software development.

IV. Related Work

Multiple academic fields have conducted research on the subject of teamwork [10], [11]. Much emphasis has been paid to the question of what actions and elements make up teamwork as well as how teamwork influences team effectiveness [12], [13] . Self-organizing, cross-functional teams have been shown to be highly productive [10] and to improve the efficiency and precision of issue solutions [14]. However, research suggests that the outcomes of such teams depend significantly on the context and on elements like the makeup of the workforce and the organization [10], [15]. According to the literature, self-management calls for trust [16], learning, adaptability [17], and team orientation and backup behavior.

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 [18]. This, in turn, can reduce rework by identifying issues and addressing them in a timely manner. The study also found that regularly updating and tracking tasks can lead to higher software quality by ensuring that tasks are completed on time and to a high standard.

For software professionals and institutional designers who work in shared environments, JIRA is a well-liked project management application [19]. Customizable Scrum boards and adaptable Kanban boards are a crucial component of JIRA. It provides total interoperability with well-known environments like Eclipse [20].

Collaboration has a positive impact on performance, leading to increased productivity, better quality, and faster delivery times [2] . Collaboration improved communication and knowledge sharing among team members, resulting in more efficient work processes and better decision-making. Additionally, the authors found that collaboration helped to foster a more positive and supportive team culture, which in turn improved team morale and job satisfaction.

This paper [1] 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. This research paper focuses on how cross-functional agile teams communicate and update their task progress in software development projects [3]. 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.

V. Implementation

To understand the design decisions completely, we need to understand the architecture of our Progress Point software first.

A. Architecture

Building an application like Progress Point warranted the use of an MVC like architecture. We built modular components,which integrate together to make the final software:

- 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 used handlebars for the front-end 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 used 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. For this project we implemented a JWT token to authenticate the API and user activities.
- Data Storage: The data storage system consists of a database where the user-generated data is stored. This included user profiles, posts, updates, likes, and reposts. We used MongoDB for storage, and leveraged it's NoSQL properties for our project.
- Caching: The caching system is responsible for improving the performance of the application by storing frequently accessed data in memory. This included data such as user profiles, Posts, and trending topics. We implemented 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 reasons for using the specific technologies and architecture are mentioned in the next section.

B. Implementation Design Decisions

The design decisions we made for the architecture of progress point are as follows:

- Handlebars: We used handlebars because it is a popular templating engine used in web development that provides an efficient way to generate dynamic HTML content. We used handlebars for its simplicity, reusability in code, and because it is a logic-less templating engine.
- Node.js with Express framework: Node.js is a powerful and flexible platform for building real-time, data-heavy applications like progress point. Its fast performance, scalability, and large community made it a popular choice for us in building web applications today. We used express framework on top of node.js to simplify the process of building our app. Its easy-to-use API, routing capabilities, middleware system, and flexibility makes it a good choice for our app.
- JWT Tokens: Building a platform like Progress Point requires a strong emphasis on security to protect user data and prevent unauthorized access.We used JWT tokens for authentication, because they are stateless, which works best for a web application, and also provide security since they are signed and encrypted, while also being scalable, since they can easily be distributed across multiple servers and scaled horizontally.
- Mongo DB: MongoDB is a document-oriented database that provides a flexible data model for storing and retrieving data. It is a schema-less database, meaning that data can be added and removed without having to define a schema upfront. This makes it easier to change the structure of the data as the application evolves.This allowed us to store data in a way that is natural to our application, making it easier to work with and scale over time.
- MVC Architecture: We used MVC architecture style to ensure separation of concerns. To maintain this separation, it's important to ensure that each component only contains code relevant to its specific responsibilities. This separation allowed us to work on each component independently, without affecting the other components. It also makes it easier to modify or replace one component without affecting the others.
- Modularity: To build a scalable and maintainable application, it was important to break down the system into smaller, modular components. This allowed different parts of the application to be developed and tested independently, and made 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 is why we buult progress point using easily scalable tools like Node.js and Mongo DB, while also including techniques like sharding the database and using caching to store relevant data in eaisily accessible caches.

C. Processes

For developing progress point, we incorporated the use of agile methodologies kanban and scrum. Agile methodology is an iterative approach to software development that emphasizes collaboration, flexibility, and customer satisfaction. Kanban is a visual workflow management tool that helps teams manage their work by visualizing tasks and limiting work in progress. Scrum is an iterative and incremental framework for managing product development that focuses on team collaboration and co-operation through regular standup meetings. Apart from these agile methodologies, we also made the use of GitHub to handle issue tracking and version management during the course of our development. We had Scrum meetings twice a week to discuss progress and plan for the upcoming sprint, and

we used Kanban to visualize our workflow and limit work in progress. This helped our team stay focused on delivering a working product, while also ensuring that we were managing our workflow effectively and avoiding bottlenecks.

1) Kanban: We used kanban as a workflow management tool to visualize work, limit work in progress, and improve overall efficiency. The board was divided into columns that represented the different stages of the workflow, which were "To do", "In Progress" and "Done". We also created columns "Tests- in progress", "Tests- Completed" and "Issues" to easily keep track of tests and issues. With the help of kanban, we were able to visualize the progress of our development, while also limiting the amount of work in progress at any given time in order to improve flow and reduce bottlenecks. This also allowed us to identify bottlenecks or tasks that were stuck in a particular column for too long, and we were able to address these issues before they became bigger problems.

2) Scrum: We used scrum in the form a stand-up meeting that took place for 20 minutes every Tuesday and Thursday before class.At each stand-up meeting, we talked about the set of features we need to work on next, while also talking about out current progress and whether there are any blockers that need to be addressed. We spoke about what aspect of the project will each team member be working on till the next stand-up, and addressed if there were any changes to the deliverables. Scrum helped us increase collaboration between teammates, and also to identify any issues or roadblocks early so that they can be addressed quickly.

D. Testing Framework

To ensure that the application is functioning as expected, we established a comprehensive testing framework. This included unit tests, integration tests, and end-to-end tests.

1) Unit Testing: We performed unit testing on individual units of our software application in isolation from the rest of the system. The purpose of unit testing was to validate that each unit of the software application is working as intended and to catch any errors or defects before they can impact other parts of the system. For this, we created automated test scripts, that ran and tested the smallest units of code, which were the individual functions inside our code. We used Mocha and Chai testing framework to write unit tests. We tested the individual files for the data that was being passed through their APIs, and also other individual functionalities of the system like the creation of a user,creation of a chatroom and loading user homepage. Unit testing helped us ensure the quality and reliability of the code. It also saved time and resources by catching issues early in the development process, before they become more complex and difficult to fix.

2) Integration Testing: Integration testing was used to test the interactions between different modules of our software system. The purpose of integration testing was to verify that the modules are working together as intended and that they are properly integrated with each other. During integration testing, we combined multiple modules of our software together and then tested them together as a group.This allowed us to identify any issued that may arise due to the way the modules interact with each other. In integration testing, we combined the login functionality and the chatroom functionality to test, whether a chatroom created by a manager is visible to the developers on searching the chat room name. We also tested to see whether a developer is able to see the posts that are posted to a chatroom that they are subscribed to, and whether they are able to like and comment on the posts.

3) Acceptance Testing: Acceptance testing is a software testing technique that is used to determine whether a software system meets the requirements and expectations that were set during the inception of the project. The purpose of acceptance testing is to ensure that the software system is ready to be deployed and used in a production environment. For acceptance testing, we tested the entire system as a whole against the initial objectives that we had laid down for the project. We tested whether a developer is able to login to their dashboard, search for chat-rooms to subscribe to as well as whether they can see, like and comment the posts that are present in the chatroom they are subscribed to. We also checked whether they can create their own posts. Similarly, for the manager, we tested whether they can create new chat-rooms, and make posts in the chat-rooms that they create, while also being able to like other people's posts and give feedback by commenting on other posts. By conducting acceptance testing, we ensured that the software system meets the needs and expectations that were initially laid down by the team.

VI. Deployment and Maintenance

Our Progress Point will be deployed using Kubernetes, a container orchestration platform, and the infrastructure will be hosted on Amazon Web Services (AWS). Kubernetes provides us with a robust and scalable environment to deploy our application while minimizing downtime during updates and scaling.

A. Deployment Plan

- Set up the Infrastructure: We will set up a cluster of EC2 instances, which will host the Kubernetes control plane, worker nodes, and the database. The database will be hosted on an Amazon RDS instance to ensure high availability, scalability, and reliability.
- Deployment Strategy: We will use a rolling deployment strategy to deploy new versions of the application. This strategy will allow us to minimize downtime and ensure that the application remains available during the deployment process. Rolling deployments will also enable us to quickly roll back to a previous version in case of issues.
- Version Control: We will use Git as our version control system and GitHub as our code repository. We will create a branching model that will enable us to keep our main branch clean and always deployable.
- Continuous Integration and Deployment: We will use a Continuous Integration and Deployment (CI/CD) pipeline to automate the deployment process. We will use GitHub

Actions to build and test the application when new code is pushed to the repository.
- Test Automation: To automate testing, we will integrate our testing framework with the CI/CD pipeline using tools like cypress or selenium. We will create automated tests to cover both functional and non-functional requirements, including performance, security, and accessibility testing. We will also ensure that tests are run at each stage of the pipeline, including unit tests, integration tests, and end-to-end tests. This will help us catch issues early in the development cycle and ensure that the application is continuously tested and verified as new changes are deployed.
- Scaling and Backup: We will configure Kubernetes Horizontal Pod Autoscaler (HPA) to automatically scale the application based on the CPU or memory utilization. We will create an automated backup process to regularly backup the data, including the database, configuration files, and user data, to an Amazon S3 bucket.
- Security: We will implement best security practices like SSL encryption, two-factor authentication, and secure storage of sensitive information like passwords and API keys. We will also use Kubernetes secrets to store sensitive information securely.
- Maintenance: We will use Kubernetes rolling updates to ensure that the application is regularly maintained with patches and updates to fix security vulnerabilities and improve performance. We will also conduct regular load testing and performance testing to ensure that the application can handle high traffic.
- Disaster Recovery: We will create a disaster recovery plan in case of a catastrophic event like a server crash, data breach, or natural disaster. This includes having backups, testing the backup and restore process, and having a plan for restoring services as quickly as possible.

B. Maintenance Plan

We will conduct regular maintenance activities to ensure that Progress Point application remains reliable, performant, and secure.

- Patch and Update Management: We will regularly apply patches and updates to the operating system, software, and dependencies. We will also keep our Kubernetes version up to date.
- Performance and Load Testing: We will conduct regular performance and load testing to ensure that the application can handle high traffic and remain performant.
- Security Management: We will conduct regular security scans and vulnerability assessments to identify and remediate security vulnerabilities.
- Backup and Recovery Management: We will regularly test our backup and recovery process to ensure that we can restore the application and its data in case of a disaster.

By leveraging Kubernetes, AWS, Docker containers, and modern CI/CD pipelines, we can deploy and maintain our application with ease while ensuring the highest level of reliability, scalability, and security.

VII. Future Scope and Limitations

Progress Point is a useful tool to facilitate real-time communication and sharing updates. Currently, Progress Point allows for creating separate chat rooms and sharing updates based on team chat. People can also react to these updates and share thoughts. While Progress Point can provide numerous benefits to software engineering teams, there are also some limitations to consider and the future scope for this application is vast, with opportunities to expand its features and capabilities.

A. Limitations

- User can create login as a Manager or Developer. Only manager has access to create new team room.
- Progress Point does not allow personal chat or group chat. User can share updates directly to the entire team room.
- Progress Point does not have a notification feature for updates or comments shared by other team members
- Currently, the system only supports images as attachment.
- User needs to search for a team room name to follow it, Progress Point doesn't have a feature to show all the available team rooms currently.

B. Future Scope

- Personal Chats: This feature can be added so that developers can interact with each other personally and get more freedom in discussions.
- Voice and Video calling: Features for direct voice calling and video calling can be added in Progress Point to facilitate ease of communication and may be helpful for urgent and quick meeting sessions.
- Whiteboard for Discussions: Integrating a virtual whiteboard into the application can facilitate real-time brainstorming and problem-solving sessions, making it easier for teams to collaborate on ideas and solutions.
- File sharing: The ability to share files and documents within the application can streamline the communication process and reduce the need for external file-sharing services.
- Mobile Applications: Many teams work remotely or on the go, making mobile applications a critical component of a successful application. By developing mobile applications that are optimized for smartphones and tablets, teams can stay connected and collaborate effectively regardless of their location.
- Integration with Third Party Applications: Integrating with third-party applications such as Google Drive, Dropbox, and Trello can provide teams with additional functionality and simplify their workflow. By enabling teams to access and manage their data and projects directly within the Progress Point application, teams can save time and work more efficiently.

VIII. Conclusion

Effective communication is essential in software engineering teams, especially in remote work environments. The lack of clear and open communication can lead to misunderstandings, missed deadlines, and ultimately a decrease in productivity. This is where Progress Point can be an ideal solution to enhance team collaboration and communication. software engineering teams can have a centralized platform that enables them to share information, updates, and collaborate on projects in real-time.

Progress Point allows users to post updates and images in team room. It also offers various features such as reacting and commenting on posts. The ability to create separate team rooms for specific projects, or topics can also help reduce noise and increase clarity in communication.

In conclusion, Progress Point application can significantly improve communication in software engineering teams. It offers a range of features that facilitate real-time collaboration, reduce noise, and increase clarity in communication. By using such an application, teams can work together more effectively, share information, and ultimately increase productivity.

References

[1] 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.

[2] 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.

[3] 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.

[4] T. Niinimäki, A. Piri, P. Hynninen, and C. Lassenius, "Studying communication in agile software development: a research framework and pilot study," in *Proceedings of the ICMI-MLMI'09 Workshop on Multimodal Sensor-Based Systems and Mobile Phones for Social Computing*, 2009, pp. 1–4.

[5] J. D. Herbsleb, A. Mockus, T. A. Finholt, and R. E. Grinter, "Distance, dependencies, and delay in a global collaboration," in *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, 2000, pp. 319–328.

[6] S. McIntosh, Y. Kamei, B. Adams, and A. E. Hassan, "An empirical study of the impact of modern code review practices on software quality," *Empirical Software Engineering*, vol. 21, pp. 2146–2189, 2016.

[7] J. W. Gilley, M. L. Morris, A. M. Waite, T. Coates, and A. Veliquette, "Integrated theoretical model for building effective teams," *Advances in Developing Human Resources*, vol. 12, no. 1, pp. 7–28, 2010.

[8] V. Stray and N. B. Moe, "Understanding coordination in global software engineering: A mixed-methods study on the use of meetings and slack," *Journal of Systems and Software*, vol. 170, p. 110717, 2020.

[9] L. Williams, G. Brown, A. Meltzer, and N. Nagappan, "Scrum+ engineering practices: Experiences of three microsoft teams," in *2011 International Symposium on Empirical Software Engineering and Measurement*. IEEE, 2011, pp. 463–471.

[10] R. A. Guzzo and M. W. Dickson, "Teams in organizations: Recent research on performance and effectiveness," *Annual review of psychology*, vol. 47, no. 1, pp. 307–338, 1996.

[11] J. Sapsed, J. Bessant, D. Partington, D. Tranfield, and M. Young, "Teamworking and knowledge management: a review of converging themes," *International journal of management reviews*, vol. 4, no. 1, pp. 71–85, 2002.

[12] C. W. Langfred, "The paradox of self-management: Individual and group autonomy in work groups," *Journal of Organizational Behavior*, vol. 21, no. 5, pp. 563–585, 2000.

[13] E. Salas, D. E. Sims, and C. S. Burke, "Is there a "big five" in teamwork?" *Small group research*, vol. 36, no. 5, pp. 555–599, 2005.

[14] J. Tata and S. Prasad, "Team self-management, organizational structure, and judgments of team effectiveness," *Journal of Managerial Issues*, pp. 248–265, 2004.

[15] S. G. Cohen and D. E. Bailey, "What makes teams work: Group effectiveness research from the shop floor to the executive suite," *Journal of management*, vol. 23, no. 3, pp. 239–290, 1997.

[16] S. Nerur and V. Balijepally, "Theoretical reflections on agile development methodologies," *Communications of the ACM*, vol. 50, no. 3, pp. 79–83, 2007.

[17] G. Morgan, "Reflections on images of organization and its implications for organization and environment," *Organization & Environment*, vol. 24, no. 4, pp. 459–478, 2011.

[18] 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.

[19] D. Özkan and A. Mishra, "Agile project management tools: A brief comprative view," *Cybernetics and Information Technologies*, vol. 19, no. 4, pp. 17–25, 2019.

[20] J. P. Rodríguez, C. Ebert, and A. Vizcaino, "Technologies and tools for distributed teams," *IEEE software*, vol. 27, no. 5, pp. 10–14, 2010.