KarmaTracker: Slack Reward Bot

Project Proposal

Morgan Pham morganpham3300@vt.edu
Aiden Barker aidenbarker@vt.edu
Eunice Hong eunice2713@vt.edu
Vatsa Bhatt vb24@vt.edu

ABSTRACT

While software engineering has grown leaps and bounds since its beginnings, there are many problems that plague the field as described by the Software Crisis. A significant piece to the problem is the development of low-quality code. The issue described in the Software Crisis is that machines have become so powerful that we no longer understand how to keep up with it. To attempt to address this growing problem, the proposed solution is to recognize and reward employees for performing actions that work to solve the low code quality problem. For teams that use Slack, this Slack bot named “KarmaTracker” would keep track of each employee’s good karma points which are appointed by other employees recognizing these good deeds. A dedicated Slack channel would be created to name the top “Karma Point” climbers each week, giving the space to acknowledge and celebrate these high performers. Management should be encouraged to connect employees’ KarmaTracker points to their performance metrics and performance reviews to get a clearer picture of who is working to improve the work of others and therefore, the overall company’s.

INTRODUCTION

Whether this low quality is deemed through inefficient code, code failing to meet requirements, or code that is poorly maintained, this problem of quality deterioration must be addressed for the field to continue growing in an upwards direction. Many of these causes can be ameliorated with simple actions such as setting up quick calls to pair program or perform debugging sessions rather than communicating back and forth over Slack, completing a code review for a peer without being instructed to, answering questions about your team’s code outside of your own team channel, etc. A strong lack of communication between one’s own team as well as cross-teams has contributed to the problem of low-quality code, as some are concerned that spending time helping others won’t be acknowledged and will only look like less time spent completing their own work. This kind of collaboration needs to be encouraged, which can be done by having a system to recognize and reward those that take the time to help their peers. Because KarmaTracker stores each employee’s points, employees can have a specific place to point to when discussing performance with their managers or whoever else may be concerned, to show that they are strong team players willing to help the overall success of the company.

RELATED WORK

GitHub Stars and Contributions is relevant to KarmaTracker as they demonstrate the effectiveness of recognition and reward systems in promoting collaboration and code quality improvement within the software engineering community. These GitHub features serve as a great example for promoting good behaviors and interactions among team members in the context of KarmaTracker. Similarly to how GitHub users may "star" repositories and "follow" developers to recognize and monitor their contributions, KarmaTracker allows users to give points to peers who do actions that improve software quality, inspire cooperation, or facilitate cross-team communication. This recognition system matches with the purpose of your project, which is to promote collaboration and communication among software developers.

Furthermore, KarmaTracker's user recognition and leaderboard functions are inspired by GitHub's contribution history, which showcases a developer's involvement and skill through their pull requests, problems, and commits. KarmaTracker encourages good contributions by recording and presenting users' points and achievements. It also allows others to evaluate their colleagues' engagement and talents. Furthermore, GitHub's involvement in supporting code reviews and issue tracking demonstrates how better communication and cooperation may result in higher-quality code—a premise that is closely aligned with the project's goals.

As this project progresses, we can draw on GitHub's success in promoting collaboration, code quality, and recognition to inform the design and implementation of KarmaTracker. By adapting and integrating these concepts, KarmaTracker can effectively encourage better communication and teamwork among software engineers, thereby enhancing the efficiency and quality of software development processes.

SOFTWARE ENGINEERING PROCESS

Our team chose an Agile software engineering approach, namely Scrum, for the KarmaTracker project. This decision is motivated by the project's aims of encouraging cooperation and communication among software developers since Scrum's iterative and adaptive approach fits in perfectly with these goals. We want to build KarmaTracker in stages, starting with essential capabilities like point tracking and user recognition. This tiered approach allows us to present consumers with a working bot sooner, allowing them to reap its primary benefits. Furthermore, it allows us to implement additional features in later sprints based on user input and altering priorities, in accordance with Agile principles that stress delivering incremental value and reacting to changing requirements. Flexible conditions like this allow us to iterate back to previous stages and continue improving features as needed.

Following development, we will emphasize deployment and monitoring, ensuring that KarmaTracker is easily accessible from our Slack workplace while also providing monitoring and error tracking systems to ensure seamless functioning. By doing sprint retrospectives at the conclusion of each development cycle, our team will be able to reflect on what went well and what may be improved in our development process, allowing for ongoing progress. Finally, we will maintain an active user feedback loop throughout development, aggressively seeking and incorporating user insights and data-driven choices to align the bot with user expectations and improve its performance. We want to provide a robust, user-centric KarmaTracker by adhering to our Agile process, which successfully handles the difficulties of low-quality code and fosters improved communication between our software engineering teams.

REFERENCES

[1] stars.github.com/program/