Reward Bot*

Encourage and Motivate students

Eric Jung
Computer Science
Virginia Tech
Blacksburg VA USA
ericjung04@vt.edu

Lakshitha Gattu
Computer Science
Virginia Tech
Blacksburg VA USA
lakshithag@vt.edu

Yoonje Lee
Computer Science
Virginia Tech
Blacksburg VA USA
ylee201@vt.edu

Chris Johnston
Mechanical Engineering
Virginia Tech
Blacksburg VA USA
chrisj25@vt.edu

Abstract

With the rise of technology we have today, students are not motivated to get their work done, as apps like social media and online shopping have made access to instant dopamine very easy. Thus, students tend to put off their work and put their attention on the things that provide them instant gratification. Our proposed solution is a system that gives random rewards to team members for performing actions (i.e. commiting code to a repo, finishing assignments on Canvas, etc.). The system can give rewards such as custom icons for your profile picture, prizes like discount code coupons, and even bonus points for completing a task early. Upon receiving these rewards, students would then be provided with dopamine that they earned for completing a task, not from scrolling on social media. This in turn would help students get over their addiction to their phones and become more productive.

Introduction

Our ultimate goal for this project is to motivate students to get their work done, which would in turn help improve their work ethic and motivation, directly correlating to an improvement in their academics, confidence, and abilities to write good and quality code. Being able to write clean and efficient software would increase student's chances of landing a software engineering internship, or job upon graduation, resulting in innovative and creative software for the future. It would focus on what affects students negatively and positively, and work on those things, as well as reinforce the good ones to increase their overall academic performance [1][2].

Related Works

While there are no distinct research studies/software tools on the matter of rewards as it pertains to software engineering, there has been a recent explosion in reward softwares for consumer use. Popular examples are Points, Walkr, Sweatcoin, StepUP, and Arrow, which take simple daily tasks or exercise and turn them into a fun game where you can get points. With over half a million combined users and an average store rating of 4.5 stars, these apps reveal that reward software is finding a foothold, making them a powerful case study for the value a tool like ours could add to the software engineering landscape. In related works, it includes how social media affects teenagers' mental health who fears missing out–[1]. Also, it would focus on rewarding students for their efforts on their work, which would create habits that are positive [2].

Motivating Example

Say we had a student studying who was struggling to stay focused, get their work done, and succeed in school. Instead, they focused all of their time and attention into video games, scrolling on social media, and hanging out with friends. While it is extremely important to maintain a healthy work-life balance, this person is not getting any work done, and only focusing on relaxing. At the moment, this may feel good and comforting, but living this kind of lifestyle could lead to further consequences. It could lead to things like lower grades, missed opportunities, and a sense of regret. So how do we solve this ever-growing issue with each new generation of students? We introduced the system, RewardBot, which provides students with rewards for submitting their work. By taking a more competitive approach to these assignments, students are then incentivized with points, to take small, and more manageable steps towards finishing their work. These points can be redeemed for prizes like extra points and coupons, which encourages students to put in effort, and offer a sense of accomplishment. This concept is not only applicable to students, but to every single person in the world, including software engineers. In the realm of software engineering, you often face tight deadlines with strict requirements, which require your full and foremost attention. Our system can be integrated into tools like GitHub or Figma so that when a new task/feature is completed, a reward is given to provide the sense of completion and gratification. This provides increments of dopamine to the developer, so that they don't feel constant friction when trying to finish their task.

Software Process

The software engineering process our team decided to use is agile because it uses iterative development which increases flexibility for our team. We are able to adjust to changes quickly since the iterative cycles are shorter. As a team, we decided that it was better for us to break the big project into smaller, more manageable pieces, which allow us to get continuous feedback from our stakeholders. The process of getting feedback through the development of the project allows the stakeholders to share their expectations for the project and discuss potential issues before they become major complications. Agile also emphasizes team collaboration by doing daily stand up scrum meetings where each team member can share their progress, setbacks and future plans. This allows the team to stay on the same page and share ideas and thoughts. Additionally, the scrum meetings will be documented which makes it easier when new members join or team members leave. The onboarding process for new members will be expedited since they have documentation to go over each person's contributions to the project.

Requirements Elicitation

For our requirements elicitation, our team decided to use the survey method. Surveys are advantageous in the early stages of development, as it is able to incorporate multiple types of questions and data eg. can have multiple choice questions, open/closed ended questions, etc). We were able to balance the questions we asked between open ended questions like free responses, and closed ended questions like likert scales. This was very beneficial to our team, as we now had data on how much users of this system would prefer to have certain functions, as well as data on current systems in place today similar to our product. While this technique was effective in capturing our needed requirements, it also has its limitations. Now that we have an initial set of data and requirements to work upon, we must now formulate new questions and surveys to conduct, which might take more time and not be as effective as other forms of requirements elicitation. This goes to show that the software engineering process is very dynamic, and oftentimes requires changes.

Design Choices

For our project, we will be using the layered architectural pattern, specifically the MVC (Model View Controller) model. To implement this, we have chosen the MERN (MongoDB, Express.js, React.js, and Node.js) technology stack. According to the official MongoDB website, MERN is a 3-tiered architecture consisting of a front-end built with React.js to create the user interface, a back-end using Express.js and Node.js to handle HTTP requests and communicate with the front-end, and a database tier, where MongoDB will be used to store and manage data [3].

We decided to use the layered 3-tier architecture because of team contribution and modularity. First, since the purpose of our project is to build an application that stores, processes, and visually displays data to users, a web application was the most suitable choice. Additionally, one of our team members has experience building web applications with the MERN stack, and React is relatively straightforward to learn and implement, allowing all team members to effectively contribute. Second, the MVC architecture offers significant flexibility and maintainability. By separating the Model, View, and Controller layers, we can independently modify or replace components without overhauling the entire system. For example, while we initially chose MongoDB for the database, it is possible to switch to MySQL or another database system with minimal changes to the rest of the application. This flexibility also extends to supporting multiple views for a single dataset, which can be useful if we expand to mobile or other platforms in the future.

Implementation Process

The implementation process of our project was divided into several phases to ensure a structured and efficient development workflow. Initially, as planned, we created a prototype to identify additional features needed based on the use cases and storyboard developed during the planning phase. This was followed by front-end development using React.js, where dynamic and responsive user interfaces were created based on the prototype. In this phase, we designed the interface with static data that did not change. Then, we planned the datasets and schemas that needed to dynamically change and be stored for the backend. In the next phase, we implemented backend development using Express.js and Node.js to handle server-side logic and establish the API endpoints. During this phase, we focused on connecting with MongoDB. Once the core functionalities were completed, we ensured smooth communication between the frontend and backend. Finally, we iterated on the process to add or remove necessary features after checking usability and testing the program ourselves, incorporating feedback as needed.

To ensure the quality of the system, we conducted comprehensive testing at multiple levels. For testing the program, we primarily used the white-box unit testing method. Since our program utilized Axios to communicate between the front-end and back-end, we first tested whether the back-end successfully retrieved data from the database and whether the front-end correctly retrieved data from the back-end. These tests were verified using assert cases tailored to our system’s functionality. Additionally, we created a demo user to test authentication and authorization processes to ensure they worked properly for security purposes. Integration testing was carried out to verify that the front-end and back-end communicated seamlessly. Usability testing was also performed to evaluate the user interface and overall user experience. After identifying areas for improvement, we iterated on the system to address any issues. This process

allowed us to refine the application and ensure reliability before deployment. Finally, feedback from stakeholders was incorporated to align the system with user expectations and requirements.

Deploy and Maintain

For future deployment, we plan to use several hosting services to ensure reliability and scalability. For the front end, we aim to use Github Pages, as these services are optimized for single-page applications and provide features like automatic HTTPS, CI/CD pipelines, and easy scaling. For the back end, we will use a cloud platform like Heroku, AWS Elastic Beanstalk, or Render to handle application hosting, scaling, and runtime environment setup. For the database, MongoDB will be utilized as a cloud-based managed database service, ensuring scalability, security, and ease of use. We will configure environment variables such as API keys and database URIs using .env files during development. To streamline the deployment process, we will use tools like GitHub Actions to automate build, test, and deployment workflows. Since part of our application involves a Chrome extension, we will package and upload the extension to the Chrome Web Store, ensuring that permissions and manifest files are correctly configured to align with Chrome's policies. Versioning will also be implemented to roll out updates without disrupting users. Additionally, we plan to explore different deployment strategies, including rolling deployment to gradually replace old versions.

Future Project Extensions

For future milestones of the project, there are several features we plan to implement. First, we aim to integrate authentication with the Canvas LMS. To achieve this, we will need to obtain developer access for Canvas, which was challenging to accomplish within our initial time frame. Currently, our program requires users to create an account with a user-generated Canvas token. With this, our program can retrieve Canvas data using the Canvas APIs and convert it. In the future, we plan to implement a direct login feature with the Canvas LMS. Second, we aim to replace static values. Even though we connected the data to the backend, some static values remain, such as the semester date range. Currently, these are hard-coded to span from August 23rd, 2024, to December 18th, 2024. In the future, we will enable users to customize these dates according to their school or college. Finally, this extension only works for Virginia Tech's Canvas page. This is because of the implementation of retrieving Canvas data using the APIs. We are currently not sure why it did not work when we implemented it; however, in the future, we plan to allow it to all Canvas users.

Conclusion

In today's technology driven world, students struggle to find motivation and stay focused when completing their work, as there are so many distractions like social media and online shopping. Our project, RewardBot, was designed to help students stay motivated and feel a sense of accomplishment by offering a unique reward-based system that gives students an incentive for finishing their work on time. By aligning the efforts of students with positive reinforcement, our group aims to create habits that will promote academic and professional success. Throughout the semester, our team was able to identify the need for any additional tools and strategies that would help students be more motivated. We explored related works like *Task for Canvas*, and from there, were able to design a system that would gamify the process of completing work. While our project is designed specifically for Canvas LMS, it has potential to revolutionize many other LMS' available today. Future work to be done is to expand its capabilities with other systems, refine any current bugs and issues, and incorporate more advanced features. Ultimately, RewardBot is a stepping stone towards addressing the issue of unmotivated students and professionals, offering a practical solution to create motivation in today's fast paced society.

REFERENCES

[1] Johnson, S. B., & Pulley, C. V. (2020). Considerations for the development of a COVID-19 vaccine. Frontiers in Immunology, 11, 1880. https://doi.org/10.3389/fimmu.2020.01880

[2] Ketterlin-Geller, L. (2023, March 6). A reward system can build a homework habit. Here's how. Education Week. https://www.edweek.org/leadership/opinion-a-reward-system-can-build-a-homework-habit-heres-how/2023/03

[3] MongoDB. (n.d.). MERN stack explained: A full-stack guide. MongoDB. https://www.mongodb.com/resources/languages/mern-stack