The FocusBot Solution

Enhancing Software Developer Productivity

Michael Pang
Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia, USA
michaelp03@vt.edu

Ajay Seethana
Department of Aerospace and Ocean Engineering
Virginia Polytechnic Institute and State University Blacksburg, Virginia, USA
ajayseethana@vt.edu

Matthew Topping
Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia, USA
mctopping@vt.edu

Elijah Tynes
Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia, USA
elijaht@vt.edu

ABSTRACT

As software engineering is currently being driven by humans, adverse human factors will affect the quality of the software being developed. As the demands on software engineers increase, the neglect of essential self-care will lead to adverse health effects, such as dehydration, fatigue, eye strain, and more. To address these challenges, we propose FocusBot, an application designed to enhance software developers' well-being and productivity by encouraging regular breaks and promoting healthy work habits. Drawing inspiration from existing tools like WorkRave and EyeLeo, FocusBot integrates productivity tracking with break reminders, offering personalized task suggestions to help developers strike a balance between work and self-care. The application utilizes event-based architecture for real-time responsiveness and scalability, ensuring timely notifications based on user activity. This paper outlines the design, implementation, and testing of FocusBot, as well as potential future enhancements like smartwatch integration and advanced data analytics.

INTRODUCTION

As software engineering remains a human-driven field, various human factors can negatively impact the quality of software produced. As demands on software engineers intensify, developers will tend to neglect essential self-care, manifesting issues such as dehydration, fatigue, eye strain, and more. Studies have shown that these negative health issues will affect the developer's performance and well-being. Numerous UK studies, as presented by Cigna healthcare, attribute poor physical and mental health to lower levels of productivity in the workplace (costing the UK economy around 180 billion dollars per year). Furthermore, the research concluded that promoting a work-life balance improved productivity in the workplace. As such, a means of improving the quality of life for software engineers is necessary to ensure high productivity, work satisfaction, and improved physical/mental health. FocusBot addresses this by providing productivity tracking with break reminders and offering personalized task suggestions to help developers strike a balance between work and self-care.

MOTIVATING EXAMPLE

As an example of how our project would work, meet Bob, a software engineer who recently started working remotely for his company. However, due to working the last couple years on-site, he found himself having issues maintaining his productivity after starting to work from home. However, instead of stopping, Bob decides to push through with the work, as opposed to taking a break, which only hurts his productivity levels more. That's where our project comes in. By checking the productivity and attentiveness levels of the user, the software can alert the user that they are having trouble staying focused and force them to take a break by locking access to the IDEs being used by the employee. Since Bob is unable to force himself to continue working, he reluctantly gets up to mow the lawn for around an hour before coming back to his desk. Surprisingly, Bob feels invigorated and ready to get back to work, which teaches him the value of taking breaks to avoid living a sedentary lifestyle, helping him build a healthier lifestyle.

Therefore, our project is relevant to software engineers, as there are a lot of software engineering-related companies moving towards allowing their employees to work remotely. Although this may seem beneficial with how employees would be able to work from the comfort of their own home, it can actually be detrimental to the productivity of the teams involved. Our project would be used in helping remote employees remain focused during the workday by making sure to provide them with ample break time throughout the workday, and to encourage them to adopt healthier lifestyles by moving around more often to avoid physical/mental exhaustion.

RELATED WORK

Relevant software engineering tools that we found while researching and developing FocusBot include WorkRave, RescueTime, Stretchly, StandApp, and EyeLeo. Each of these tools is specifically geared towards reminding software engineers to take breaks from their work, which is one of our main purposes for FocusBot. For example, Stretchly and StandApp notify the users to take breaks throughout the day, reminding users to take breaks to maintain a healthy lifestyle.

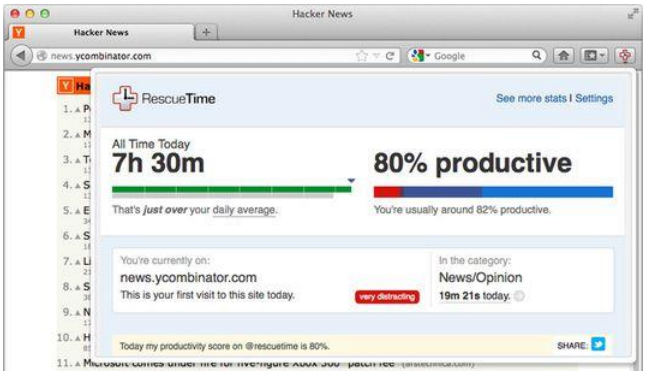


Figure 1: RescueTime interface

Another software engineering tool that was relevant to our implementation of FocusBot was WorkRave. WorkRave is a program that assists in the recovery and prevention of different injuries such as Repetitive Strain Injury (RSI), Carpal Tunnel Syndrome, and myopia. This is relevant to our project, since one of our objectives is to help software engineers avoid worsening their physical health by reminding them to avoid a sedentary lifestyle.

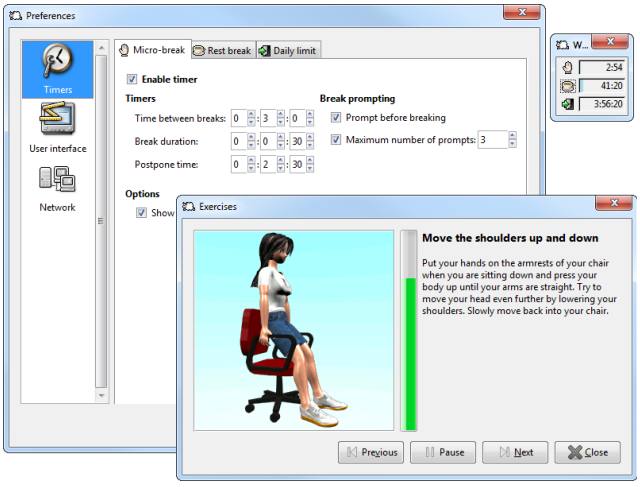


Figure 2: WorkRave interface

In a study by Cigna Healthcare in 2023, it was discovered that "the total cost to the UK economy of lost output among working-age people due to ill health is around $180 billion per annum, equivalent to 7% of GDP – productivity is by far the most significant aspect … lack of productivity is almost twice as costly as the others combined… these costs are rising – in fact they have risen by around 60% since 2016" (*Healthy employees are more productive*). This demonstrates how important it is for employees to take breaks throughout their schedule to avoid declines in physical and mental health, supplementing our goals for this project.

Overall, our project would be novel solution to the issue of software engineers losing productivity and facing declines in physical/mental health as a result of remaining sedentary for long periods of the time during the workday, because it takes inspiration from several existing applications and adds on its own unique twists. For example, our application combines the productivity tracking of EyeLeo with break reminder functionalities from applications like WorkRave and StandApp. Additionally, instead of just reminding people to take breaks, our application aims to give the users tasks that they may want to complete. For example, users can edit the notification system to add tasks that the user means to get done,

like chores. This would allow them to complete the tasks that would otherwise be pushed back by work. In between these periods of chores, the application would still provide the default suggestions that prioritize the user getting up and moving around, like stretching and doing mini-exercises, since sometimes the user should be given the time to just relax during the exhausting workday.

HIGH-LEVEL DESIGN, IMPLEMENTATION, AND TESTING

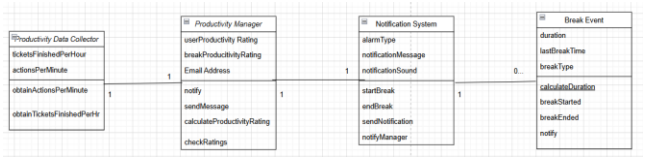
For the high-level design decisions, FocusBot utilizes event-based architecture. Event-based architecture holds the advantage of real-time responsiveness, which greatly benefits our application, as it is able to collect the data about the user's productivity (rate at which tickets are completed, how active their IDE, etc.) and respond in real time when the application notices any drops in productivity or prolonged periods of working. Therefore, the real-time nature of event-based architecture allows the application to effectively collect and analyze user data, allowing it to react immediately when the application deems a break to be necessary for the user's well-being.

Additionally, the event-based architecture also provides fault tolerance, which is highly beneficial for our application. For example, when a user is notified about an incoming break, they will also be notified about a potential break activity they can do during that time that will take them away from their seats to avoid being too stagnant during the workday. However, if this component fails or becomes unresponsive, we wouldn't want the break notification system, which would be handled by a separate entity, to also go down. Therefore, by using the event-driven architecture, the application would be able to remain operational even when one of the components is not working properly.

Event-based architecture also enables the application to take advantage of the asynchronous processing of events and have the benefit of scalability. Asynchronous processing of events allows events to be supervised at the same time, improving the overall efficiency of the application. Scalability allows our application to be utilized by a wider audience of employees at the same time. These are both particularly important components, since we don't want the application to run into issues collecting data/triggering events at the same time for several company's worth of employees.

Another reason to apply event-based architecture is the ability to integrate external services that would expand the functionality of the application. For example, companies may want to store the data collected by the application in a data storage for use in meetings like sprint retrospective meetings, where managers may want to reflect on the past sprints and the performance of each of the employees/teams.

To implement the FocusBot application, the Agile software engineering process was employed in addition to the requirements engineering process. This included the strategy of requirements elicitation, requirements analysis, requirements specification, and requirements prioritization.



Prototyping and feature mock-ups were used to depict the core functionality of our application.

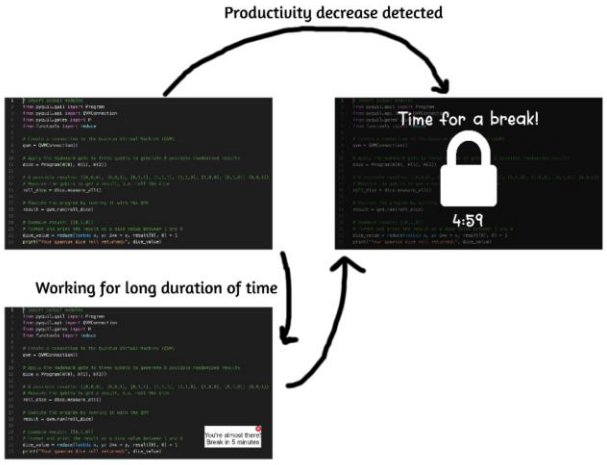


Figure 4: FocusBot feature mock-up

As for testing strategies employed, black box testing was used to validate the functionality of the application's various features. This allowed us to test the functionalities of the application by making sure that the I/O behavior matches what we expect. Additionally, by using validation testing, we were able to test the entire system to ensure that the typical product interactions of a client would work as expected.

	Test ID	Description	Expected Results	Actual Results
1	Productivity Tracking Validation Testing	Preconditions: Users have tasks to complete Steps: 1. User maintains various levels of productivity 2. Observers monitor the productivity rating of the user based on the data being obtained by the application Input: Eye-tracking data, Key-stroke data, productivity on JIRA tickets	Productivity dips when inactive and increases when active; when the productivity drops below a threshold, the user will be notified	
2	Access tasks posted on JIRA through the application Validation Testing	Preconditions: Tasks assigned to the test user on a test JIRA account that allows users to move the tickets amongst the different columns and update their progress Steps: 1. User moves task from backlog to In Progress 2. User moves task from In Progress to Completed Input: JIRA task data	Users access the test tickets through a JIRA embedding that allows them to change progress on different tasks	

Figure 5: Summary of black box testing results

DEPLOYMENT AND MAINTENANCE INCLUDING CLASS CONCEPTS

We plan to deploy and maintain our project using the canary deployment strategy. This is because it holds the lowest amount of risk amongst the deployment strategies discussed in class, meaning that it has lower potential for any prolonged/serious issues relating to system downtime, data loss, application functionalities, security vulnerabilities, and disruptions to business operations. Additionally, it allows us to deploy and maintain updates for the application efficiently, as we are able to make sure the application's update works properly with a small number of users that are actually able to interact with any new functionalities coming with the updated version of the application before releasing it to other groups. This gives us the opportunity to make sure there aren't any weird bugs in the update through constant user interaction. Introducing the update in small batches also makes it easier and faster to roll back in the case of issues than other strategies, since only a portion of the users would have to be reverted to the older version, which would still be supported for use by the other groups of users.

CONCLUSION (LIMITATIONS AND FUTURE WORK)

FocusBot represents a step forward in addressing the challenges software engineers face in balancing productivity and health. However, the development and deployment of FocusBot are not without limitations.

Limitations

- **Privacy**: Collecting user productivity data, including metrics like eye-tracking and keystrokes, raises valid concerns about how this data is stored, used, and secured. Maintaining user trust will require robust security and transparency with our privacy policies.
- **User Differences**: It is nearly impossible to employ a "one size fits all" system. Software engineers vary greatly in their work habits, health needs, and schedules. This is why we strive for highly customizable scheduling and notifications, a feature that may add complexity to the user experience.
- **Notification Fatigue**: Frequent alerts can overwhelm users, reducing their effectiveness. Designing an adaptive notification system that learns user preferences and balances reminders with productivity needs is crucial.
- **Measuring Success**: Measuring the success of FocusBot is complex. Success metrics such as increased productivity, improved health, and user satisfaction require careful tracking and analysis over time, making evaluation both resource-intensive and time consuming.

Future Work

- **Smartwatch Integration**: Incorporating smart watch technology will enable more

precise tracking of physical activity, hydration, and even stress levels like heart rate, blood pressure, etc.

- **Long-Term Pattern Recognition**: By leveraging machine learning, FocusBot could analyze historical data to identify trends and offer personalized productivity and health recommendations per developer.
- **Eye-Tracking Technologies**: Implementing eye-tracking can provide valuable insights into eye strain and focus levels, allowing for more targeted break reminders.
- **Expanded Data Integration**: Collaborating with popular project management tools (e.g., JIRA, GitHub) to provide seamless task tracking and enhanced reporting features.

FocusBot's vision is to create a healthier, more sustainable workplace for software engineers. By addressing these limitations and pursuing these enhancements, FocusBot can evolve into a comprehensive solution that not only boosts productivity but also prioritizes well-being in the software development industry.

REFERENCES

[1] Anon. Healthy employees are more productive. Retrieved September 27, 2024a from https://www.cignaglobalhealth.com/eu/your-health-plan/productivity/healthy-employees-are-more-productive

[2] Anon. Improving software developer mental well-being and productivity. Retrieved September 27, 2024b from https://www.informatics.uci.edu/8098-2/

[3] Anon. Stack overflow developer survey 2022. Retrieved September 27, 2024c from https://survey.stackoverflow.co/2022/#developer-profile-demographics

[4] Hovancik, J. (no date) Stretchly, Stretchly - The break time reminder app. Available at: https://hovancik.net/stretchly/ (Accessed: 17 December 2024).

[5] The stand up reminder: The standing desk timer you need (2022) Stand App. Available at: https://standapp.biz/ (Accessed: 17 December 2024).

[6] Take a break and relax (no date) Workrave. Available at: https://workrave.org/ (Accessed: 17 December 2024).