PM2 Submission

Michael Pang, Elijah Tynes, Matthew Topping, Ajay Seethana

Process Deliverable I

Project Goal

FocusBot aims to improve software developer productivity and well-being by monitoring health metrics and providing timely reminders for essential self-care such as hydrating and stretching. The system aims to reduce burnout and boost productivity by addressing issues like dehydration, fatigue, and eye strain. Since our team is following the Prototyping Model within Agile, our submission includes an informal prototype of the core features planned for FocusBot. The prototype demonstrates the essential functionality to address the project's main goals.

1. Prototype Overview

- Our prototype focuses on key features, including:
- Break Reminders: Notifying users to take breaks at regular intervals to reduce fatigue.
- Hydration Tracking: Alerts for users to stay hydrated throughout the workday.
- Movement Suggestions: Prompts for users to stretch or walk, promoting physical activity during work.

2. Feature Mockups

- The prototype includes basic mockups that visually represent the primary interfaces:
- Smart Watch Integration: Notification screen to monitor activity and to show reminders for hydration and movement, with easy-to-read prompts.
- User Dashboard: A simplified layout for tracking reminders, hydration levels, and user activity.

3. User Flow Demonstration

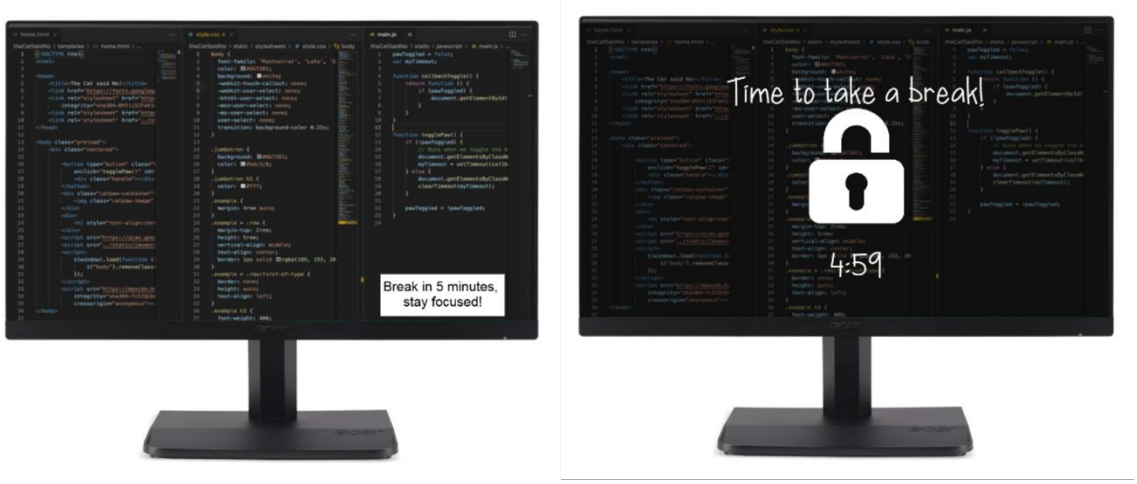
- A sample user flow outlines the sequence of events:
- Detection of Inactivity - Prompt User to Take a break - Follow-Up Notification if Inactive.
- Hydration Period - Prompt User to Drink Water - Follow-Up Prompt for Completion

4. Requirements Prioritization

- The initial features are prioritized based on our research and focus group feedback. We prioritize:
- Health reminders to prevent burnout.
- Ease of use and minimal intrusion during work.

5. Feedback Collection Plan

- After presenting the prototype to potential users, we will gather feedback on:
- Effectiveness and overall timing of notifications / customizability.
- User interface design and usability. This feedback will help to guide iterative improvements in line with our Agile development.



Requirements Analysis

Non-functional Requirements

1. Usability:

Options for settings that influence how the application works will be clearly documented underneath so that users will not accidentally make changes that they do not intend to make.

2. Reliability:

Accidental errors in which the software attempts to set a break for the user that has been set as a block that the user is busy will be fixed ASAP by giving the user the opportunity to add the period to their "blocked" time schedule with the click of a button.

3. Performance:

The application will be prioritized to be active during the workday, during which it will be active at least 90% of the time.

4. Supportability:

The application should allow easy configurability for the user settings, allowing the user to set the frequency and amount of time taken per break. Additionally, they should be able to physically add their schedule via manual additions/file importing so the application does not activate during periods at which the user is required to be online, like during meetings, presentations, etc.

5. Implementation/Constraints:

Our implementation will be created primarily through JavaScript for web application development, Swift for iOS app development.

Functional Requirements:

1. Productivity Tracker

2. Split up tasks (potentially connected to Jira for access to tickets)

3. Do not Disturb/Meeting mode

4. Ability to customize the reminder functionality (choose specific times that will vary in length based on the time from the previous break, different notification sounds for different reminders)

5. Automatically start up the system during the workday and turn off afterwards as to avoid the user needing to remember to turn it on/off per day

Use Case 1 - Productivity Tracker

Main Flow:

Application will track the productivity of the user [S1]. Productivity will be tracked using the amount of time spent on completing tickets and how active the user is while at the computer [S2]. A score will be given to the user per period of time to praise/give advice to the user in terms of how well they've done over that period [S3].

Subflows:

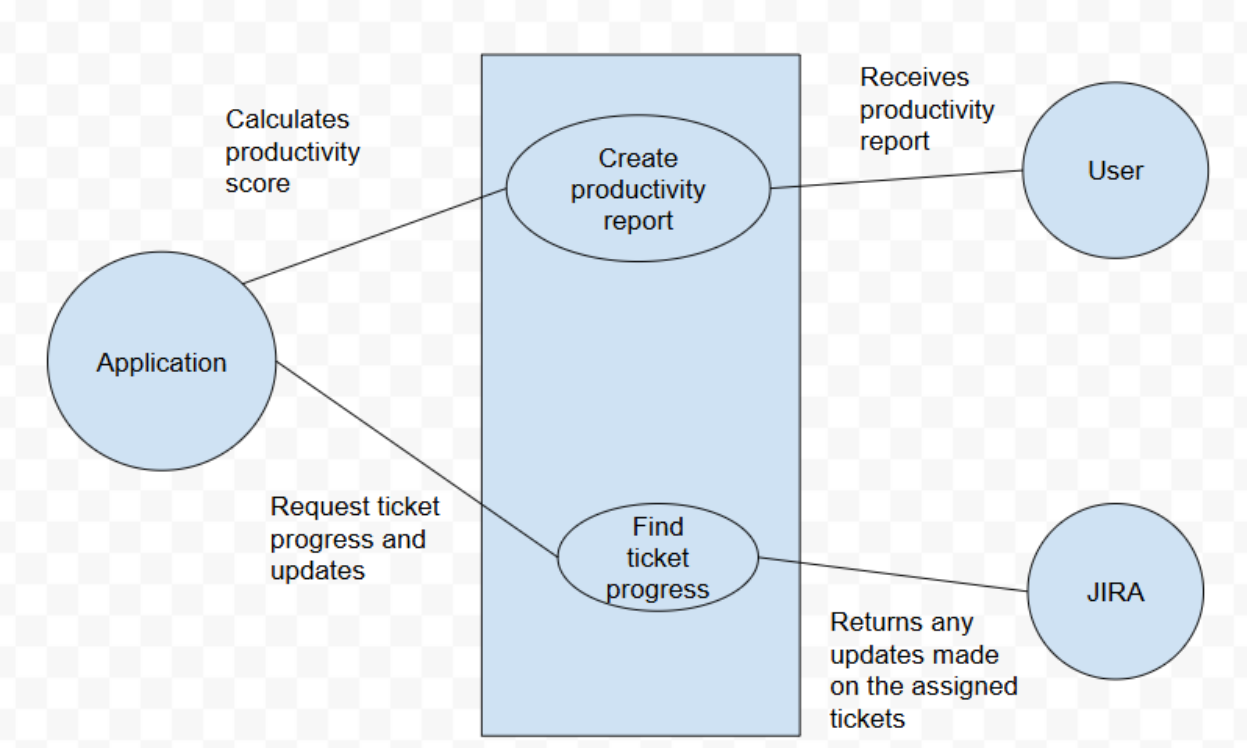
[S1] Application will track the productivity of the user

[S2] Productivity will be tracked using the amount of time spent on completing tickets and how active the user is while at the computer

[S3] A score will be given to the user per period of time to praise/give advice to the user in terms of how well they've done over that period

Alternative Flows

[E1] User cannot turn off or block the productivity tracker



Use Case 2 – Access and split up tasks (potentially connected to Jira for access to tickets)

Main Flow:

Users will be able to access, update, and split up tasks assigned on Jira through the application [S1]. The user will be able to assign time periods to each subtask [S2].

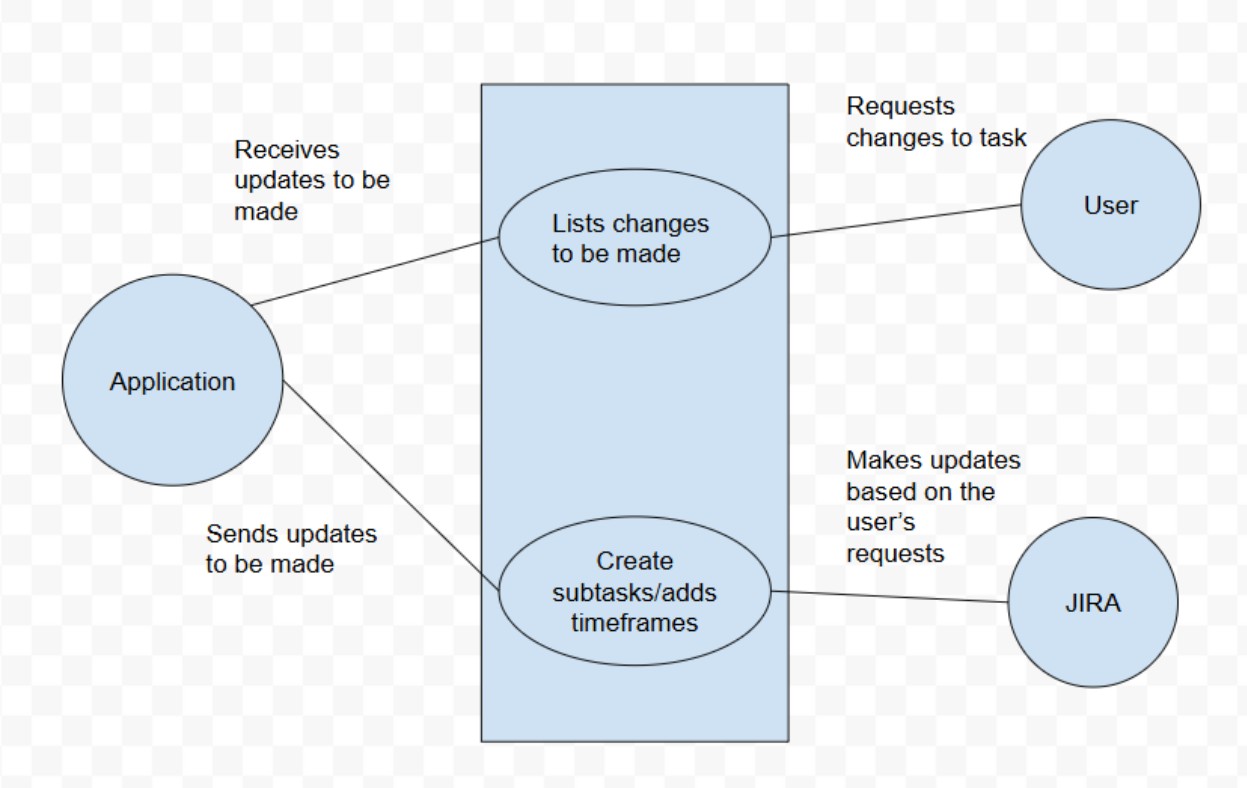
Subflows:

[S1] Users will be able to access, update, and split up tasks assigned on Jira through the application

[S2] The user will be able to assign time frames for them to complete each subtask

Alternative Flows

[E1] User is unable to assign time frames to completed tasks



Use Case 3 – Ability to customize the reminder functionality

Main Flow:

Users will be able to choose specific times that will vary in length based on the different amount of time from the previous break [S1]. There will also be different notification sounds that can be set for different notification types [S2].

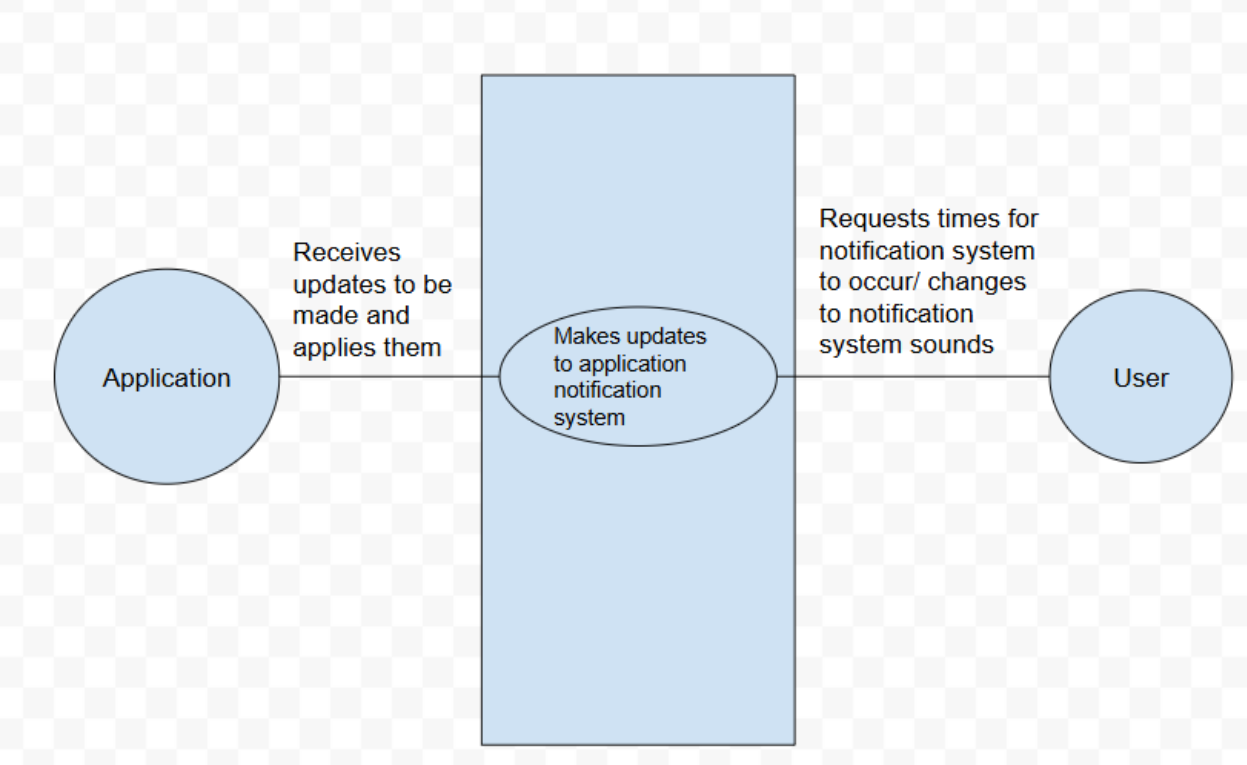
Subflows:

[S1] Users will be able to choose specific times that will vary in length based on the different amount of time from the previous break

[S2] There will also be different notification sounds that can be set for different notification types

Alternative Flows

[E1] Users will have a limit to how long the break can be (Users cannot set the break time to the whole workday, nor will they be able to set no breaks)



Use Case 4 – Do not Disturb/Meeting mode

Main Flow:

Users will be able to set a do not disturb/meeting mode that will prevent the application from sending breaktime notifications [S1]. The mode will automatically turn off after a certain amount of time to prevent the user from just turning off all break notifications [S2]. The mode will have a temporary cooldown period after use before it can be used again with a limit to how much time that can be used per day (resets per day) [S3].

Subflows:

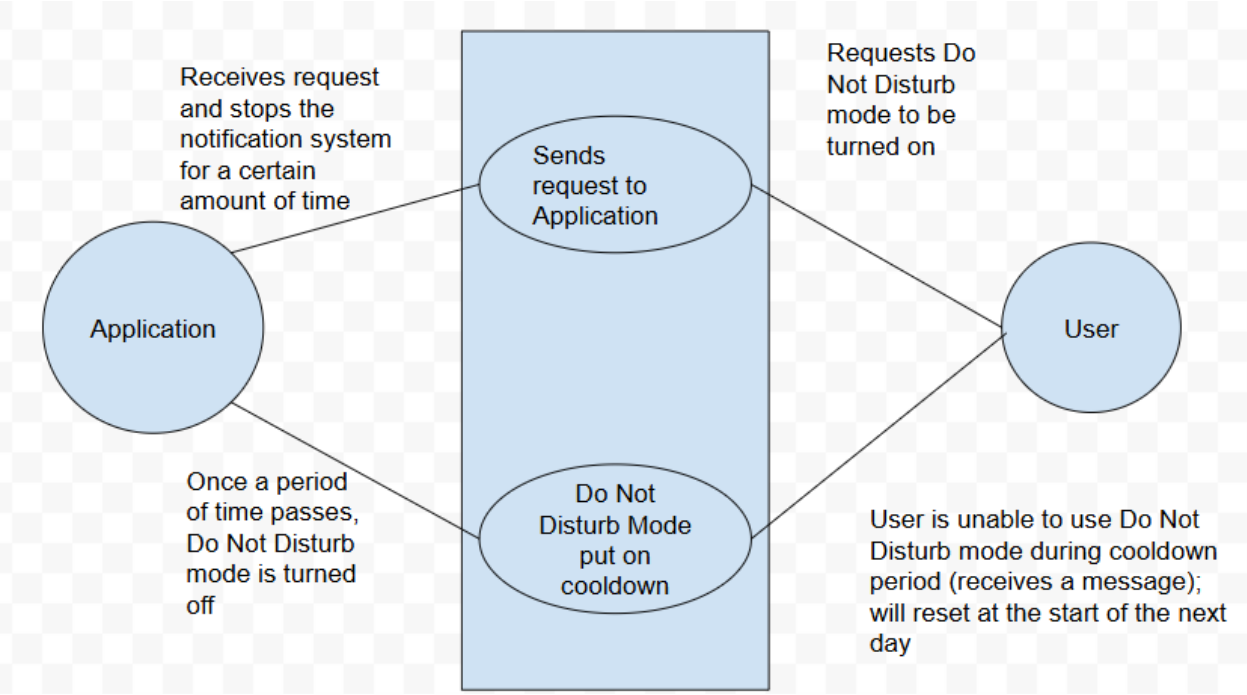
[S1] Users will be able to set a do not disturb/meeting mode that will prevent the application from sending breaktime notifications

[S2] The mode will automatically turn off after a certain amount of time as to avoid the user just turning off all break notifications

[S3] The mode will have a temporary cooldown period after use before it can be used again with a limit to how much time that can be used per day (resets per day)

Alternative Flows

[E1] User is unable to use the mode in consecutive periods during the cooldown period



Use Case 5 – Application automatically starts up/shuts down

Main Flow:

The application will automatically enable itself at the start of the workday [S1]. Additionally, it will shut itself down once the workday is over [S2].

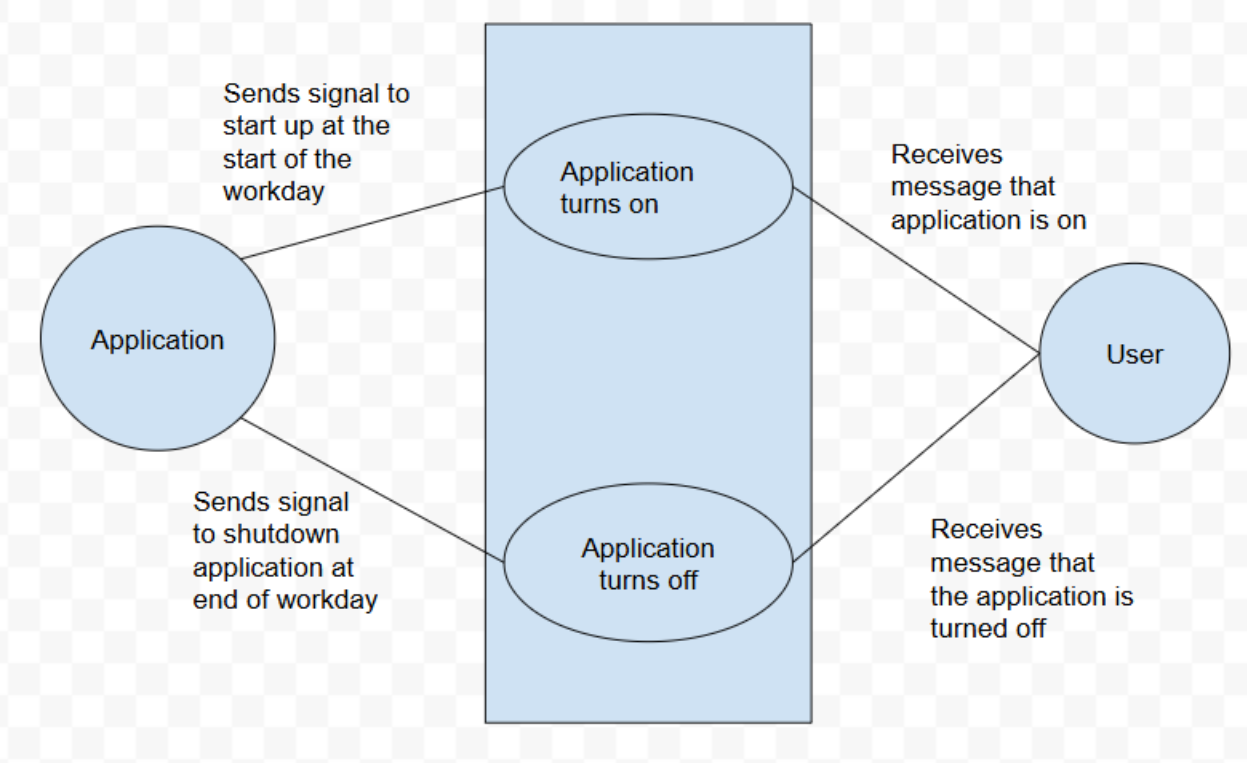
Subflows:

[S1] The application will automatically enable itself at the start of the workday

[S2] Additionally, it will shut itself down once the workday is over

Alternative Flows

[E1] Users are unable to edit the times at which the application will turn on/shut down to avoid users taking advantage of the system



Requirements Specification

User Stories

1. As a User, I want to turn on Do-not-disturb mode, so that I can focus for a long period of time without interruption.
    a. Acceptance Criteria:
        i. Given: User has no exceeded Do-not-disturb limit
        ii. When: User clicks on Do-not-disturb button
        iii. Then: System turns on Do-not-disturb
    b. Function Points:
        i. Counter for Do-not-disturb limit – 1 function point
        ii. Binding Do-not-disturb to button click – 2 function points
        iii. Activating Do-not-disturb – 1 function point
2. As a User, I want to be able to take my assigned tasks on Jira so that I can plan out how long I want to work on them
    a. Acceptance Criteria:
        i. Given: User has tasks on Jira
        ii. When: Plan button is clicked
        iii. And: planned times are inputted for each task

iv. Then: System will take tasks from JIRA and associate each with the time inputted by user and remind user about tasks needing to be done

b. Function Points:
    i. Using JIRA API to grab tasks – 3 points
    ii. Binding plan to button – 2 points
    iii. Time input box – 1 points
    iv. Associating tasks with time – 2 points

3. As an admin, I want to be able to set the limits for Do-not-disturb and breaks, so that the limits match the company policy
    a. Acceptance Criteria:
        i. Given: The admin has administrator privileges
        ii. When: The admin changes the values for limits
        iii. Then: The system applies those new limits for all users
    b. Function Points:
        i. Limits input boxes – 1 point
        ii. Applies limits- 2 points
4. As an admin, I want to be able to set the application start and end times for all users so that the application is always running during the workday
    a. Acceptance Criteria:
        i. Given: The user has administrator privileges
        ii. When: The admin changes the application start/stop times and which days the application runs
        iii. Then: The system applies those new limits for all users
    b. Function Points:
        i. Start time input – 1 pt
        ii. End time input – 1 or
        iii. Applies limit – 2 pt