Project Milestone 3
Segfaulters

Isha Devgan, Michael Kopczynski, Stuti Shah, Anushka Trivedi

Process Deliverable II

Waterfall: updated SRS document with high-level, low-level, and UI design constraints. (SRS in attached PDF)

Design Sketch

Provide a brief rationale (no more than 1 paragraph) explaining some of the design decisions for your program based on the provided sketch. Use concepts discussed in class to justify your design:

(Drawing in attached PDF)

Many of the design choices we made for the PomBot interface were based on the usability heuristics we discussed in class. We have designed the interface in a way that minimizes memory load by visually presenting only relevant information of the screen that can easily be navigated to. Additionally, in order to have an **aesthetic and minimalist design**, we use a clean layout, with minimal distractions and focus on the essential elements of the PomBot. We have also intentionally ensured users are informed about their current status throughout the Pomodoro Session by displaying the timer during sessions and providing notifications when the user should switch to a break or a session ends. This was a design choice we made to align our chrome extension with the usability heuristic, **visibility of system status**. Finally, we have included a page 'PomBot Help' that includes frequently asked questions from users, a guide to using the PomBot effectively, and the original inspiration of the tool to provide users with sufficient **help and documentation** in case they struggle with the tool.

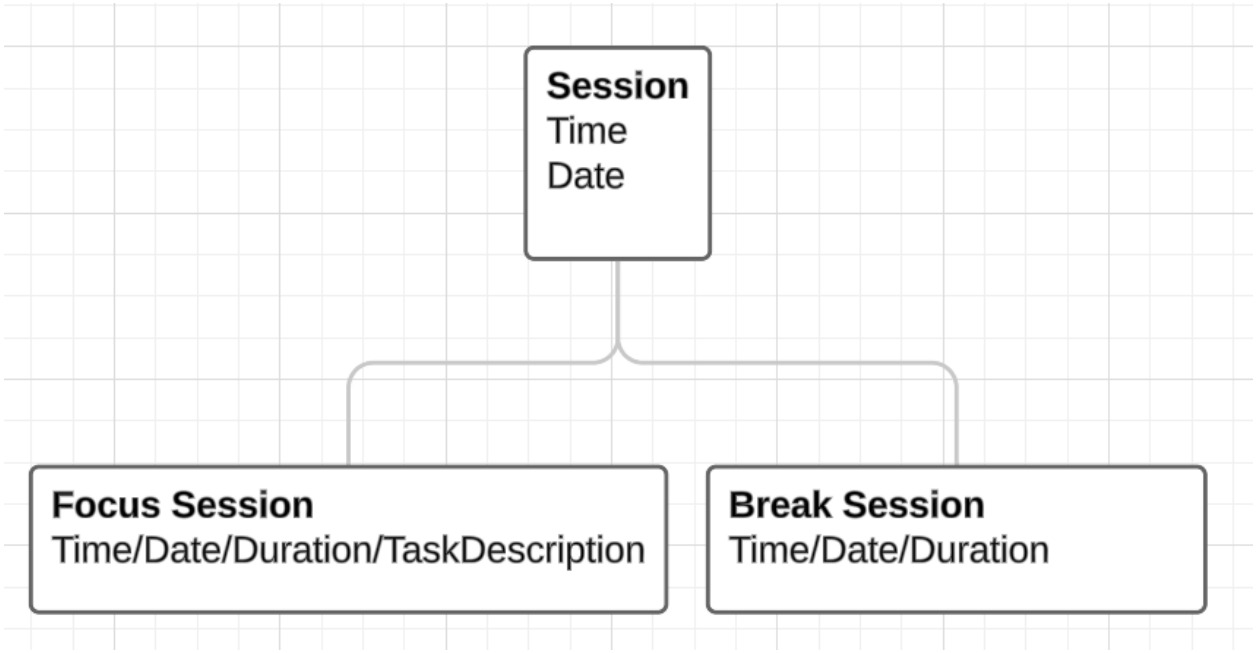
High-level Design

For a Pomodoro bot, a **2-layer Client-Server Architecture** is a good choice. In this setup, the client is the app or interface where users control the Pomodoro timer, and the server handles the tasks like tracking time, storing user data, and sending reminders. The client talks to the server through simple connections to manage the timer. Another option is the 3-layer Model-View-Controller (MVC) design, where the Model handles the main tasks (like timing and user settings), the View shows the user interface (like the timer display), and the Controller manages the communication between the Model and View based on user actions. Both designs keep our architecture organized and make it easier to maintain and improve the system.

Low-level Design
Discuss which design pattern family might be helpful for implementing a specific subtask for this project. Justify your answer, providing a code or pseudocode representation and an informal class diagram:

A design pattern family that might be helpful for implementing a subtask of PomBot, more specifically, the ability to be able to view past focus sessions, would be the **"Creation" type family**, and more specifically, the Builder. This is because it would allow for different focus/break sessions on PomBot to be accounted for. For example, you can have a class be dedicated as a "session." More specifically, you can have the sessions be specified to account for whether it is a "break" or a focus session. Based on the different object creations, it can then be put together for a summary that employers/employees can use to effectively track work within the time period the specify. See the pseudocode example below.

TotalSessions[] = [FocusSession1 on 11/11/24 for 32 minutes, BreakSession1 on 11/11/24 for 5 minutes, FocusSession2 on 11/12/24 for 45 minutes]
FocusSessions[] = [FocusSession1 on 11/11/24 for 32 minutes, FocusSession2 on 11/12/24 for 45 minutes]
BreakSessions[]=[BreakSession1 on 11/11/24 for 5 minutes]
SessionRequest = Focus sessions between 11/11/24-11/11/24 // returns the one focus session from 11/11
SessionRequest = Break sessions on 11/13/24 // returns none because there are no recorded breaks on 11/13



Project Check-In
Completed the survey to provide an update on your team progress on the project for this semester.