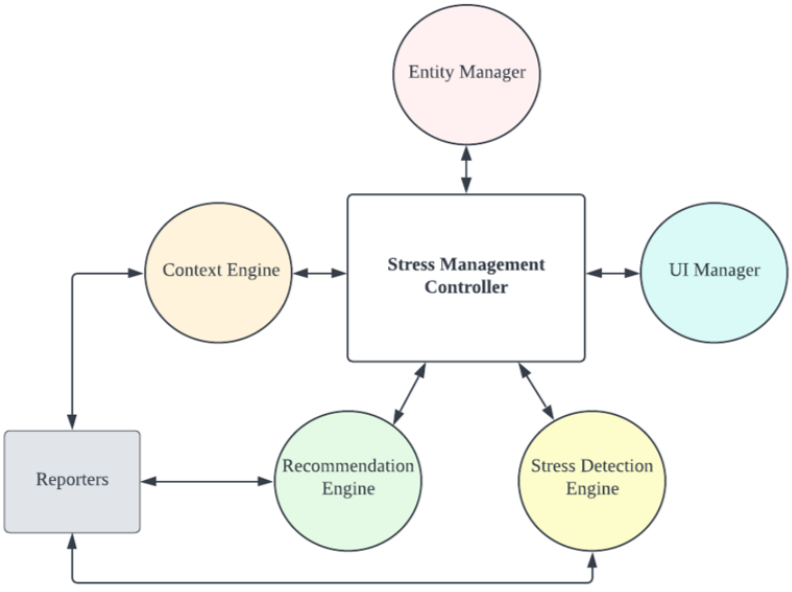
De-Stress Me: Brief Overview

William Ingold, Geonsun Lee, Irtaza Shahid, Yoon Kyung Shon, Hyemi Song

Stress is an escalated state of a human body in response to challenging conditions. We all face stress during our daily lives: while working on some critical task, thinking about some future scenario, worrying about some problem etc. Short-term exposure to stress might increase productivity for a short duration, but constant stress can cause mental and physical disorders such as cardiovascular diseases, hypertension, diabetes, cancer, headaches, depression, anxiety, and insomnia. Considering these drastic effects of stress, we are envisioning a personal digital assistant that can constantly monitor a user's stress level, and based on the user's context and detected stress level, be able to recommend some activities or actions that can help in reducing/controlling the stress level.

In this project, we are proposing **De-Stress Me**, a system that can continuously monitor stress levels and provide appropriate recommendations to regulate the stress levels. The architecture of De-Stress Me comprises the following 6 modules: Controller, Entity Manager, Context Engine, Detection Engine, Recommendation Engine, and User Interface. Figure attached below shows all the modules present in De-Stress Me's architecture.



1. **Controller**

In order to support asynchronous performance for each module and scalability of the system, we implemented a controller to manage work flows between all engines and managers. First, the Controller generates all the engines that it needs for stress detection flow. After the controller gets a message back from all engines, it starts periodic detection and recommendation procedures. Finally the controller sends result data to the front-end, so that users can be notified with current stress levels and get appropriate treatment at the exact point of time.

2. **Entity Manager**

   Recent development of IoT wearable devices enables them to provide not just one but various kinds of health data to an end user. Therefore at the end user side, collected data from devices can be overlapped and excessive. To prevent this problem, the controller gets knowledge of system-connected device lists from the entity manager and sends this information over to the context engine to sort out only necessary data for stress detection

3. **Context Engine**

   In order to track the current context of the user, we have created the StressContextEngine. This houses a range of what we have called reporters, which report on the current context and environment. Each reporter is a single actor responsible for a single type of contextual information. The StressContextEngine offers a way to abstract away interfacing with numerous types of entities and allows focus to remain upon the data, rather than how it is obtained. This allows a simpler interface to interact with for both the detection and recommendation engines.

   De-Stress Me is currently designed to operate with the following reporters: Blood Pressure, Heart Rate, Sleep, Media, Location, Scheduler, Busyness, and Medical History. Each reporter is tied to a specific entity, which it relies upon for its respective data (heart rate, blood pressure, etc.). These reporters operate asynchronously, querying their entities for pertinent, fresh data. References for these reporter actors are shared with the rest of the system. This way any who may require specific data may directly communicate with the respective reporter.

4. **Detection Engine**

The purpose of this module is to estimate the stress level so that it can be used to provide appropriate recommendations to the user. For this purpose, we are relying on 5 different measurements (heart rate, blood pressure systolic, blood pressure diastolic, sleep hours, and busyness) that provide the contextual information about the user. Then we have trained a multi-class logistic regression model to predict the current stress level in 5 different discrete levels.

5. **Recommendation Engine**

   To suggest users context-appropriate stress relief treatments, our recommendation engine takes two factors into account; the detected stress level of the user and the current status of the user. First, we set up a treatment database with predefined treatment recommendations. Each treatment considers three conditions; stress level, user's sleep status, and user's current location.

   The workflow of the recommendation engine is as follows: With the determined stress level from the detection engine, the controller passes on relevant information to the recommendation engine including the user's past and current stress level. Once the recommendation engine receives the stress level from the controller, it first determines whether the stress level has changed from the previous message. If so, the engine calls reporters that provide the environmental information of the user's status. Such information is critical to elect treatments that can be performed by the user immediately.

6. **User Interface**

   User Interface comprises two parts: back-end and front-end. The back-end is structured into three key segments: StressWebserver, Stress Webhandler, WebRoutes. The job of the StressWebserver is listening for new HTTP connections and serving said clients. The StressWebHandler and WebRoutes form De-Stress ME's API and other web routes. The front-end side of De-Stress Me is implemented as a web-app using React Redux. On the landing page, Dashboard, the user can view their real-time physiological data including heartbeat rate, blood pressure, and sleeping hours. A pop-up window will be displayed when there is a change in the detected stress level. The user will proceed to see a stress-relief treatment recommendation to resolve their stress. To reinforce our detection and recommendation engine, we also attempt to collect user's reaction to the presented information.

Conclusion

In this paper, we presented De-Stress Me, a personalized stress detection system that proposes context-based treatments which correspond to the detected stress level. Utilizing the actor model, the framework of De-Stress Me is designed in a way so that it is **scalable**, multiple devices and entities can be attached to the system, and **extensible**, additional contexts can be added or contexts collected from the system can be used in a different stress-related system. For stress level detection, we used a total of 5 features to define the level of stress and achieved 95% of accuracy by using logistic regression. Based on the detected stress level and additional contextual information for the user's current status, an appropriate treatment is recommended and pushed to the \name application's UI.