De-Stress Me: Developer Guide

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

Software Code

Our code is hosted on two separate GitHub repositories.

The main implementation for this PDA aspect is located here: https://github.com/wingold-student/cmsc818g-pda (CMSC818G-PDA). In this repository you will find a Java based project, requiring JDK 18 and Maven 1.8 in order to build and run. The JDK must be at least this version due to the use of new features, such as lambdas, which the Akka framework relies upon. Any additional dependencies will be taken care of by running “mvn install” while compilation and execution can be done with “mvn compile exec:exec” within the root directory of the repository.

Additionally, the back-end utilizes a machine learning Python script. Python 3 is required, along w with the following libraries:

- pandas
- numpy
- sklearn

The user interface repository is located here: https://github.com/HyemiSong/PDA-App It was developed separately out of ease and because it is React-Redux based, running on its own server as well. However, it interfaces with the RESTful API that is hosted in the project from the main repository.

Instructions on How To Use It

The Main De-Stress Application

The CMSC818G-PDA requires only one line to run it (after being installed), “mvn compile exec:exec”. This will kick off the entire application, which begins within “src/main/java/com/cmsc818/App.java”. An Akka actor system is launched, whereby the StressManagementController.java is the root of this system. All other actors are then launched hierarchically in an automatic fashion.

Rather than relying upon interactions with real entities and devices, we’ve built the data around SQLite databases. If you’d like to inspect the data, it is within “src/main/resources/EntityDB.db”. Reporters then read from their respective tables within this database.

For flexibility's sake, we've implemented YAML configuration files for the engines and reporters. These are located within the "src/main/resources/configs" folder. Within these configs, you can change what database is being utilized and the reading rate (in seconds) that a reporter will read a row from its respective table. By default it will be 5 seconds.

While running, all pertinent (and potentially extraneous) information is printed out to standard output through Akka's wrapped of the SL4J logger. This output will generally include which actor has output data along with its message, or give status updates about Akka systems.

The RESTful API web server will be run at "localhost:8080". If you'd like to see data, you can query the "localhost:8080/data" endpoint. Refreshing it will show new data from the detection and recommendation engine, and general reporter information.

The Front End

The front-end requires both NodeJS v16.15.0 and yarn to be installed. Within the root directory of this repository you must run yarn install followed by yarn start. Once the server is up and running, it will be hosted at "localhost:3000"

The dashboard displays current readings from De-Stress Me. Whenever the stress level elevates a pop-up will appear with a recommendation status. From the pop-up a user can read the recommendation or dismiss it. If opening the recommendation, we provide a guide for a potential treatment and allow (although not currently implemented) the user to rate this treatment.

There is also a list of all recommendations and treatments offered by De-Stress Me with their respective ratings. This page is meant to allow the user to add, remove, or update the recommended treatments manually when desired in order to improve their experience with the application.

Lastly, also not hooked up yet, is the user's profile. Here the user could potentially enter their current status if desired. This could be useful for when peripheral devices and entities may not correctly pick up the user's current mental status.