<3

Heartrate Behavior and

Analysis Tool (HBAT)

Sprint 1 Retrospective

Ruhana Azam, Manoj Polisetti, Rajith Weerasinghe, Phillip Thain

Team 3

# What went well?

**As a researcher, I would like to input data in a form of a CSV file.**

* A Parser class was made which takes in the file paths to either a Data Grid file or a combination of RR Data and Behavioral Data CSV file and and parses the files to make their respective lists of data. Then, a helper method takes in both the lists and the user inputted variables of RR\_SyncTime, BEH\_SyncTime and RR\_StartTime to calculate the relative time shift required to align both the files and create a HashMap which can be sent to the Algorithm class for the application of the algorithm.
* There is a window within the GUI that allows the user to input a file from the dialog window. The path of the selected file is then placed into a textbox, where the user may choose to change the path or filename.

**As a researcher, I would like to export results in CSV files.**

* Once the data is analyzed, the user has the option to export the file. If the user chooses to export the data, the data is sent to a class which will create a .csv file. This class will write the data onto the file in CSV format line by line for a single dataset. The file will be created in the path that is specified by the user.

**As a researcher, I would like to be able easily access the program through a graphical interface.**

* After some research, the primary library used in to create the GUI is JavaFX, chosen for its look and feel, ease of development, and integration into common IDE’s such as NetBeans. However, it does not have as large a community as Swing, the primary other alternative.
* A layout for each page has been developed for each of the different windows. In addition, they are able to interact with the user and other pages. However, there are certain pages that will change and evolve as the project goes on. For example, the data view window does not currently have a widget that displays the graph of the output. Also, the main page will expand as functionality of the main program leads to more windows being developed.

**As a researcher, I would like to easily sync up the data inputs to start at the same time stamp.**

* The Parser class has helper methods which take in the List’s of RR data and Behavioral data and the user inputted times for RR\_StartTime, RR\_SyncTime and BEH\_SyncTime to calculate the offset of the times to create an absolute time scale which then syncs the Behavioral and RR data together to create a HashMap with the absolute time as the key and the Attribute associated as the Value of the map.

**As a developer, I would like to organize information with classes**

* Aften our project owner sent us a standardized data file, we came up with a standard way to convert the data into objects which made data organization easier for the developers. We decided to store all data in hashmap for the efficiency where the the timestamp is is the key and the Attribute class is the data for that time. The Attribute holds two types of data: Behavioral and HeartRate. A class was created for each type of data. A trial.java class was created to store the hashmap and all data related to the trial. Child.java was created to keep track of all the trial the child is involved in. Group.java was created to store a list of trial.java and info related to a trail. This can later be used for batch processing. In addition, most of these classes have overloaded constructor for more effective usage for developers.

What did not go well?

Although we were able to complete a majority of what we set out to do during this Sprint, we underestimated some of our tasks, forcing us to leave some functionality behind, described below. Not to mention, it took a fair amount of time to get everyone on the same page in regards to how to use github, and setting up work environments. In addition, we learned some lessons about how poor logistics can really halt workflow. For example, we had questions about some test files we received, but we did not plan meetings with our project owner far ahead enough to be able to meet with her.

Following are the tasks and user stories that were not fully complete after sprint one. We plan to implement these tasks with priority during the next sprint.

* **As a researcher, I would like to process heartbeat data in order to analyze it against behavioral data.**
  + This task was mostly complete, however it does not meet all of our expectations and is fully complete only based on assumptions.
  + Completed: A class has been implemented that has two child classes: one will calculate the heart rate baseline/rrChange and the other will calculate the phases. To calculate the baseline, the class will assume that all the data is in sequential order and in a certain format. The phases are calculated off the baseline so once the baseLine has been calculated the data is passed to the class calculating the phase.
  + Incomplete: A few assumptions were made about what the user should input into the program. However, we are awaiting some confirmation from our project owner about those, and that could cause the parser and algorithm to change slightly.
* **As a researcher, I want to see the output file in spreadsheet form from within the GUI window.** 
  + Due to poor communication, the class that contains the data to be placed into the spreadsheet was not compatible with the way the JavaFX TableView class takes data in. By the time this issue was discovered, there was not enough time to convert the one from the other.
* **A note on unit testing**
  + Although the majority of our user stories were completed, we did not set aside time to thoroughly testing them. Next sprint we will focus on making our current work more robust by creating unit tests, as well as asserting the program against test cases during every build.

How can we improve?

Our team’s productivity can be improved by holding more regular meetings. Last sprint, as a unit, we were not always necessarily on the same page about everything, and that was simply because we did not get together and talk about our successes and challenges often throughout the sprint. Under the same line of thinking, our work could have improved with more streamlined communication to help each other to stay on task throughout the week.

Another way to improve our work in the next sprint will be honing our ability to plan ahead. For example, towards the end of sprint 1, there were a few questions that we would have liked answered from the project owner about some functionality of the program. However, we were unable to meet her at a short notice, so we had to make some assumptions on our own that may not have been correct. If we had been able to plan better, we may have been able to get a meeting together to get her input instead of taking extra time to come up with assumptions that may not be entirely accurate.