OVERVIEW

This document outlines the general thought process that our group took when designing and implementing the various classes.

NEURESET DEVICE

Mediator/Observer. The NeuresetDevice class acts as controller for the overall system, though it is a tiered architecture. NeurosetDevice owns both the sessionLog and EEGHeadset owns its own session instance as well as 21 electrodes. Electrodes send signals up to EEGHeadset, where data is processed, when the EEGHeadset has useful information, like the latest average baseline or the competition of a measurement it sends a signal to the Nearest Device class to handle final implementation and the next required calls.

The neuroset device in this application handles much of the work being done. Since the neuroset device is something that can be used regardless of whether or not a PC is present, it was represented in such a way. Other classes simply call on the neuroset when requesting information, such as the session logs or date and time.

Based on our research, the Neureset was also capable of functioning without the help of a PC, and we took this into consideration when determining where most of the calculations would be done. As a result in our simulation, all calculations are done within NeuresetDevice, EEGHeadset and Electrode without the need for external devices like PC or the main UI.

The electrode class takes the waveform and creates a Qt chart and opens it up to visually represent the values. The electrode class also is capable of generating the different types of brain waves as well as adding random noise to properly simulate the general inconsistencies in the human brain.

The measurement function such as the startMeasuremnet will begin measuring the baselines frequencies and then sent to the neureset class to start the next part of the process.

PC

Utilises an MVC behavioural design pattern in order to display, and update the session history saved to the PC class. Here the model is represented by the datamodel class, the controller is represented by the pcwindow and the view is represented by the UI. The controller interacts with any other class

needed for logic here, and in this case, the PC class. Whereas, the model interacts with our database class using the dbmanager.

Our decisions on the PC class were to treat the PC as its own separate device, represented by the separate UI window. The interaction between the neureset class, where most of the data is being calculated, is done through the PC class where it contains a pointer to the instance. We felt that for the sake of a proper simulation, it would be best to have a PC class handle the input from the neureset and then input this data into the relational database.

For the UI, we had the model represented by the DataModel class in which it would take data from the sql database and update the view as the PC class added new data. This allowed a layer of abstraction, and made it so that queries could be handled by a single class instead. In addition, we thought of adding a new dialogue box that would pop up when new data arrives. This allows the user to also be able to decide on whether or not they would like to save the data or discard it.

We have implemented a checkbox in order to simulate the connection aspect of the requirements needed for the transfer data to occur. This will either disable or enable the transfer button. As the neureset device can act without the help of a PC, and thus would likely need some physical connection, this was the closest method to simulating that.

Graphical User Interface

Observer pattern and, technically, is a controller.

In our design, the GUI plays a key role as the interface through which users interact with the Neureset device and it is different from the UI seen in the PC class. While it's a component of the Neureset, the GUI is designed to fulfil more complex roles than a PC's UI, which is to be developed alongside the Neureset device. The GUI does less data processing and handling and more displaying of information processed by the Neureset device similar to the PC class. The GUI acts more as an observer or controller in this context, receiving and displaying information processed by the Neureset device.

This interaction is through signals and slots. Signals and slots serve as the primary communication method between the GUI and the Neureset device. When specific events or actions occur within the Neureset, such as data processing completion, signals are emitted. These signals are then connected

to corresponding slots within the GUI, triggering actions such as updating the display with the processed information.

In addition, we felt that the GUI should contain the PC class and the NeuresetDevice class as it works as an execution point for our program. This allows the PC class to be capable of viewing the neureset device, and extracting the session logs to save.