Textual Explanation - Team 11

We have implemented a program using Qt in C++ that simulates the functioning of a Cranial Electrical Stimulation (CES) device similar to Oasis Pro. This includes the execution of critical features such as power on, power off (forcefully turning the device off before the session ends), ending a session (soft off), detecting battery level, selecting session group and type, performing connection test, adjusting intensity and recording therapy and adding it to treatment history.

Running the program from Qt in the 3004 course virtual machine presents the user with a detailed and interactive Graphical User Interface (GUI) with different Qt elements that helps simulate the different features of the device and observe the functioning in real-time. The program has three concrete classes, the MainWindow, Session and Record. Among these, MainWindow is the most complex which handles a wide range of functionality. The most critical behavioral components of the program can be seen in this class.

While the MainWindow.ui contains the UI components such as QPushButton for all the buttons, it is the MainWindow class itself that has all the handlers registered for the UI components which extensively uses slots and signals. Additionally, it also uses key Qt components such as QList, QProgressBar, QAbstractSlider, QListWidget and QTimer to build features such as duration list, session list, progress bar and many more. This class is also responsible for initializing the values for power on, connection, record, battery level and contains the current state of the program (i.e off, select, connection, run and end phase). The MainWindow supports the key functioning of the device such as updating time labels and making per second updates to the battery etc. with the help of the Record class that it also initializes. Record is another important class that stores and sets the time duration the user selects for the therapy. Additionally, this class also stores and updates the value of the intensity attribute. It also contains a Session object and together using these values, it builds the therapy session information that can be recorded if the user chooses to at the end of the therapy. The Session class represents one particular session type. This essentially sets the name and frequency of the session type and is also responsible for returning these values. It is the interaction between these three classes that drives the program.

The design details and execution of the program is explained further in detail below.

**** Important: Please note that each of the sections below consist of key design decisions and highlights important differences between the implementation of our program and the real device and any extra features added.**

<u>USE CASE 1: TURN THE POWER ON</u>

To turn the device on, the user has to press (left click) and hold the ‘ON’ button for a few seconds. Just pressing the ‘ON’ button does not turn the device on, it must be held for a few seconds. This button is the representative of the power button on the actual device. Before the device is turned on, all the interactive buttons and labels are disabled and cannot be interacted with. This simulates the behavior of the actual device where the user cannot use the device unless it is first turned on. Once the user releases the ‘ON’ button on the GUI, the device powers on and all the other buttons and screens on the GUI become enabled. The text on the button now

changes from ‘ON’ to ‘OFF’ to represent that the same button will be used to turn the device off since the actual device just has one power button that fulfills both of these purposes. This button is implemented via QPushButton. If the device is left idle for around 45 seconds then it turns off.

USE CASE 2: TURN THE POWER OFF

To turn the device off, the user must press and hold the ‘OFF’ button for a few seconds. The user may turn the device off at any point while the program is running, and it will stop the program from running. It will disable all buttons and screens to simulate the behavior that the device can no longer be used once the device is turned off. The session also ends immediately. The device also turns off when the battery is critically low as described further in the explanation. As mentioned earlier, if the device is left idle for around 45 seconds then it turns off.

USE CASE 3: END A SESSION

Whenever a session naturally ends (the duration of the therapy session reaches zero), the device goes through a soft off. Going through a soft off causes the device’s QProgressBar to scroll from 8 to 1 and enter the end phase. In the real device, this is done via the LED lights instead. The device also enters the “end” phase and does a soft off when the battery is critically low.

USE CASE 4: DETECT BATTERY LEVEL **(**CONTAINS KEY DIFFERENCES BETWEEN OUR PROGRAM IMPLEMENTATION AND ACTUAL DEVICE).**

The implementation of detecting battery level is slightly different from the real battery detection done in the actual device. Firstly, the implementation of our program has replaced the LED lights with a progress bar built using QProgressBar. Additionally, in the real device, the battery, intensity and connection are all represented via the LED lights during different points of the therapy. However, instead of displaying the battery as a part of the progress bar (which represents the LED lights), we have implemented it separately for making the device user-friendly and easier to understand. The GUI has a separate battery icon on the top left that shows the current battery level as percentage instead of separate bars. Unlike the actual device where the battery is only displayed for a few seconds when the device is turned on, and then periodically during the therapy, we instead display our battery throughout the program for better monitoring.

As the user starts the therapy session, the change in battery level can be seen via this battery icon. If the user increases the intensity level, which is described further below in the description, then the battery starts to drain more rapidly, and the user can notice more visible changes in the battery icon. The rate at which the battery depletes is based on how the user changes the intensity as the session is running. This simulates the feature that the battery percentage lost after a session is a function of therapy duration and intensity. If the connection is changed to ‘No Connection’ which is also further described below while the session is running, then the battery stops draining to represent that the therapy has been paused, and then once the connection is changed back to ‘Okay’ or ‘Excellent’, the battery starts to drain once again. This demonstrates that battery depletion is also a function of connection to skin. The battery also does not drain when the device is powered off.

After reaching low battery level (30%), the battery display turns yellow, and upon reaching critically low (10%), the battery display blinks red and turns the device off momentarily after a "soft off". If the user turns the device on when the battery is critical, the device's battery blinks red and switches back off. In order to 'change the battery', the user just has to close and restart the program from Qt, since there is no feature in place to "recharge" the battery.

USE CASE 5: SELECT SESSION GROUP AND TYPE **(**CONTAINS KEY DIFFERENCES BETWEEN OUR PROGRAM IMPLEMENTATION AND ACTUAL DEVICE).**

Once the device has been turned on, the power button can be used to select between the session groups (duration of the therapy) similar to the real device during its "selection" phase. The user just needs to press the power button and not hold it for a few seconds. The user cannot just press on an element inside the list of session types to change them, but instead has to use this button for navigation. We have implemented the three durations: 20 minutes, 45 minutes, and user designated time. User designated time can be essentially any time duration selected by the user. Though the actual session is in minutes, our program runs for the corresponding seconds, for example, if the user selects 20 minutes, the program will actually run for 20 seconds. The user designated feature has been implemented via a spin box which is different from the real device. The spin box allows the user to increase or decrease time using the arrows in the spin box that essentially allows the user to enter a custom time. This spin box is only displayed after switching to the "user defined" session group during the "selection" phase.

The UP and DOWN buttons, which are QPushButtons, can then be used to navigate between the four different types of sessions, which are: Delta, Theta, Alpha, Beta. Since the session types we decided to implement all have the same CES pulse type, we have excluded storing or displaying this property from our implementation, but we do store their corresponding frequency values. Instead of implementing icons for the session groups and types, we have displayed them in a QListWidget for greater transparency between the user and device. This is done to simulate the actual functioning of the device. As the user navigates these lists, the highlighted session group and type keeps changing, similar to the real device.

Once the user is done selecting the session group and type, the 'Select' button needs to be pressed to start the therapy. At this point the program enters the connection phase as described further below in the document and the session begins after this connection test is over. We have also implemented a timer that counts down as the session starts running, a feature that was not displayed to users in the original device. This is an extra feature that we have added using QTimer, allowing the user to keep track of the duration the session is running for and how much time is left. The timer starting is also a way to tell that the session has started successfully after a connection test. The timer reaches zero as the session ends.

USE CASE 6: PERFORM CONNECTION TEST

The connection test feature is implemented via a QAbstractSlider with labels that denote what the connection of the device is set at. In the actual device, the LED lights are responsible to show the current connection status during the connection test phase. Instead, we have used a QProgressBar to replace the LED lights. Once the user presses the 'Select' button during the

"selection" phase, the program enters the connection test state for five seconds similar to the actual device and defaults to "excellent connection". It is only after these 5 seconds of the connection test that the therapy session starts and enters the "running" phase (can be seen as the timer starting and battery starting to drain), so long as the connection is either okay or excellent. When this connection test is over, the progress bar returns back to showing 1 and now the progress bar will represent intensity (described below) once more. This is how we implemented the progress bar to display both connection status and intensity, similar to the LED lights doing both.

During a connection test phase, if the connection is "Excellent", the progress bar shows 3 and the L R icons are lit green; if the connection is okay, then the progress bar shows 6 and L R icons are lit yellow; and if there is no connection then the progress bar shows 8, the L R icons are lit red. The session is paused if there is 'No Connection' and only resumes if the connection is changed to "Okay" or "Excellent".

Entering a connection phase only happens when the user presses the select button (before the therapy session has actually started) or when the QAbstractSlider is used to change the connection to 'No Connection' while the session is running. Any changes between the "Okay" and "Excellent" connection while a session is not in its "Connection" phase will not reflect on the GUI.

USE CASE 7: ADJUST INTENSITY

Once the user presses the 'Select' button during the "selection" phase, the user can now use the 'UP' and 'DOWN' button to increase or decrease the intensity level similar to the real device. The intensity level is displayed in the progress bar which represents the LED lights of the actual device. The intensity values are 0 to 7 but the progress bar shows values 1 to 8. By default, the intensity is set at 0, which is displayed as 1 on the progress bar. As the intensity is increased by one unit, the progress bar also changes by one unit. For example, when the intensity is 1, the progress bar will show 2. The intensity can be changed at any point while the session is running. The intensity level selected will affect the rate of battery depletion as described above.

USE CASE 8: RECORD THERAPY AND ADD TO TREATMENT HISTORY

All selections and adjustments are done in the Record class via the Record pointer in the MainWindow which has been described in the beginning of the textual explanation. A record is created upon the beginning of a new session. At the end of the session, the user can choose to record the session by clicking on the 'Record session' button, which can be done until a new session begins. This will add the Record pointer to the QList<Record*> which in turn will be displayed on the GUI. There is a small white window implemented via QListWidget on the right of the GUI (on the left of the 'Record session' button) that displays the recorded therapies that have been added to the treatment history. This includes details such as session name, frequency, session duration and the most recent session intensity before the session ended. A session can only be recorded if the session ends naturally, not if the session ends due to battery being critically low or if the user forcefully turns the device off in the middle of the session.