COMP 3004: Group Project

Ali Hadi Mazeh, Hajar Alyemni, Hamza Faour, Sofia Alfonso Ortega
April 12, 2023

Table of Contents

Use Cases

Use Case 1: Normal Operation

Primary Actor: User
Goal in Context: Achieve higher coherence in heart-rhythm pattern
Precondition: User has access to device and knowledge on how to operate it
Postcondition/Success Guarantee: User achieve higher coherence
Main success scenario:

1. **User:** turn on the HeartWave
2. **User:** attach sensor to earlobe
3. **User:** navigate menu and enter session mode
4. **HeartWave:** display HRV graph, breath pacer, and metrics
5. **User:** press selector button to initiate new session
6. **HeartWave:** begin reading user heartbeat
7. **HeartWave:** calculate HRV
8. **HeartWave:** calculate coherence (use our own data)
9. **HeartWave:** update LED light colour according to coherence level (low=red, medium=blue, high=green)
10. **HeartWave:** update breath pacer value
11. **HeartWave:** Loop through steps 7-9 until user termination
12. **User:** Stop session by pressing selector button
13. **HeartWave:** calculate percentage of time in different coherence levels (low, medium and high)
14. **HeartWave:** calculate average coherence
15. **HeartWave:** calculate achievement score

16. **HeartWave:** display session summary that includes percentage of time in different coherence levels (low, medium and high), average coherence, length of session, achievement score, entire HRV graph.

Extensions:
8a. New coherence level is reached
    8a1. Beep goes off to indicate a new coherence level is reached

Use Case 2: Interruption due to Sensor Off

Primary Actor: HeartWave
Goal in Context: interrupt current session when sensor turns off
Precondition: user is currently in session
Postcondition/Success Guarantee: current session is interrupted when sensor turns off
Main success scenario:

1. **Sensor:** turns off/malfunctions
2. **HeartWave:** disable symbol to indicate loss of active pulse reading
3. **HeartWave:** end current session

Use Case 3: Battery Low

Primary Actor: HeartWave
Goal in Context: Warn the user the device is low on battery and will be shutting down
Precondition: The device is on and the user is currently using the device
Postcondition/Success Guarantee: user alerted of battery status
Main success scenario:

1. **Battery:** reaches 20% charge
2. **HeartWave:** display warning message warning user of low battery
3. **Battery:** reaches 0% charge
4. **HeartWave:** automatically end session if a session is currently active, and power off

Use Case 4: Accessing a Session From Session Logs

Primary Actor: User
Goal in Context: view previous session information
Precondition: device is on and user has previously used the device and initiated sessions
Postcondition/Success Guarantee: user is able to view previous session information
Main success scenario:

1. **User:** select “view session history” from menu via selector button
2. **HeartWave:** display all previous sessions in the form of rows
3. **User:** select specific session

4. **HeartWave:** display all information for the chosen session

Use Case 5: Changing Breath Pacer Settings

Primary Actor: User

Goal in Context: Change breath pacer setting

Precondition: Device is on

Postcondition/Success Guarantee: User is able to change breath pacer settings

Main success scenario:

1. **User:** select "Settings" from menu via selector button
2. **HeartWave:** display device settings
3. **User:** select new breath pacer setting
4. **HeartWave:** update device settings to reflect the new setting

Use Case 6: Restoring Device to Initial Install Conditions

Primary Actor: User

Goal in Context: Restore the HeartWave to its initial install conditions

Precondition: Device is on

Postcondition/Success Guarantee: User is able to reset the device to initial install conditions

Main success scenario:

1. **User:** select "Settings" from menu via selector button
2. **HeartWave:** display device settings
3. **User:** push reset button in settings screen
4. **HeartWave:** reset all settings to their default value

Use Case 7: Turn On Device

Primary Actor: User

Goal in Context: Turn on the HeartWave

Precondition: User has a HeartWave and the HeartWaved is powered off

Postcondition/Success Guarantee: the HeartWave is turned on and is on the Main Menu screen

Main success scenario:

1. **User:** Press Power Button
2. **HeartWave:** turn on

Use Case 8: Turn Off Device

Primary Actor: User

Goal in Context: Turn off the HeartWave

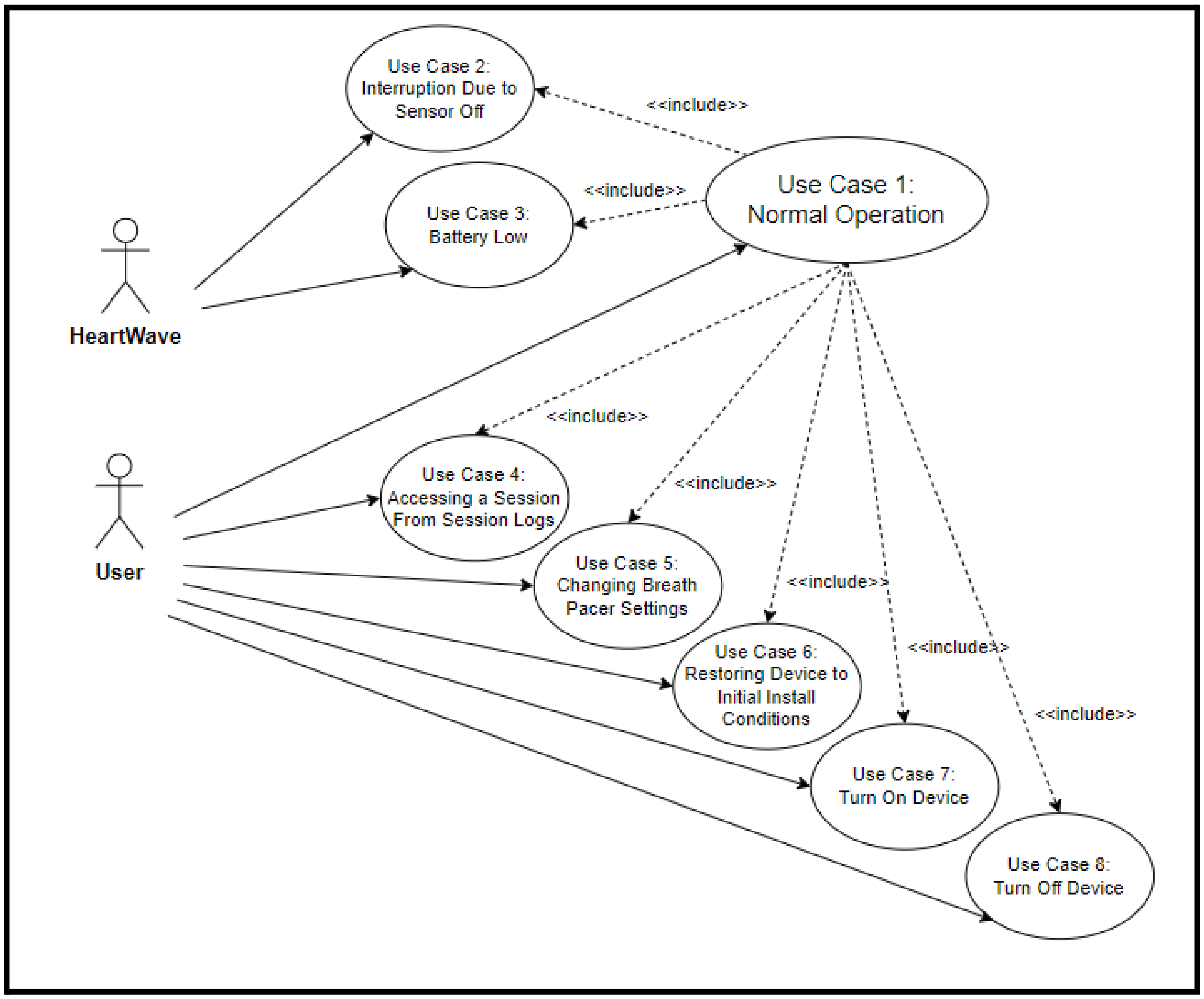
Precondition: User has a HeartWave and the HeartWave is powered on

Postcondition/Success Guarantee: the HeartWave is turned off and the display is powered off

Main success scenario:

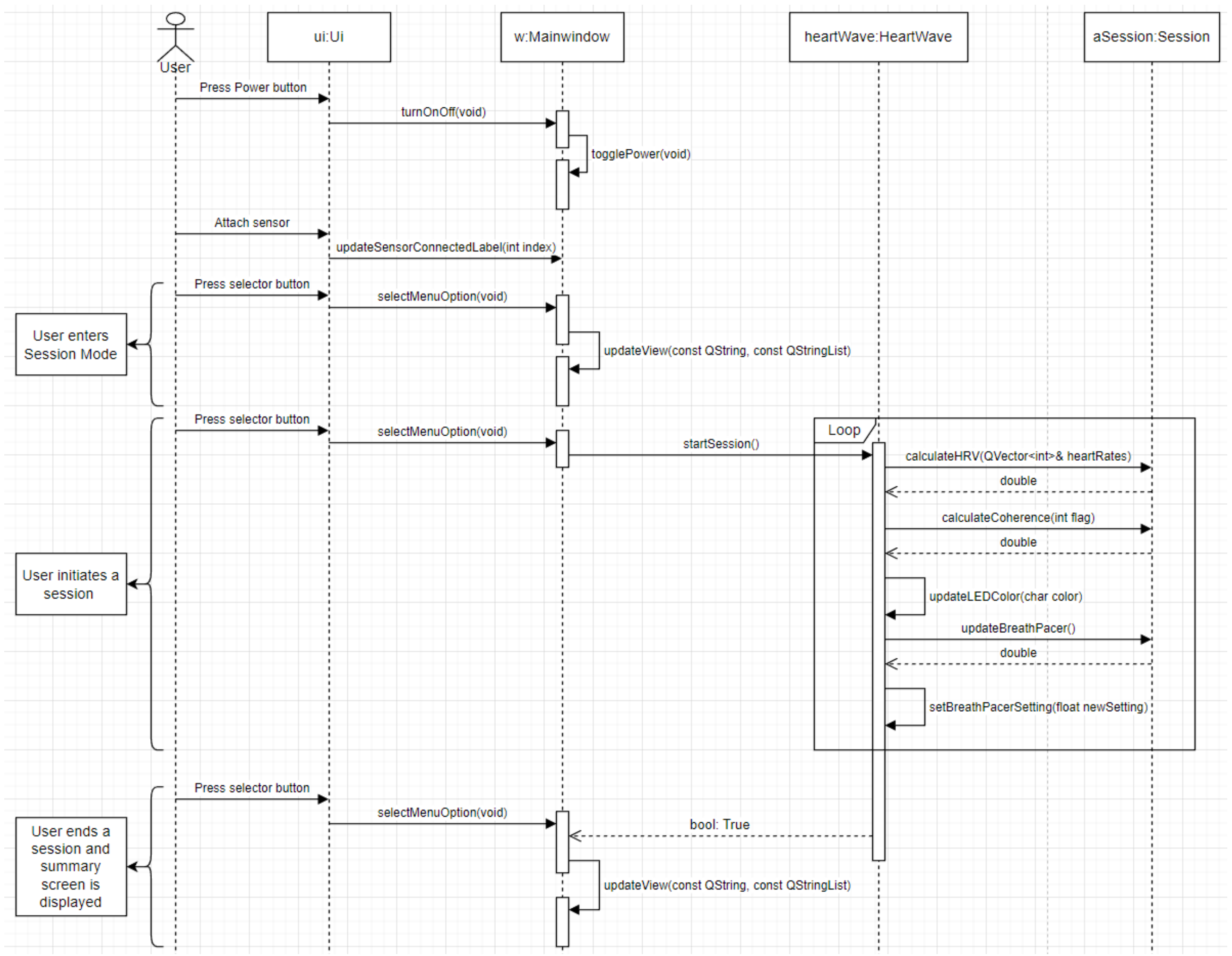
3. **User:** Press Power Button
4. **HeartWave:** Power off display
5. **HeartWave**: Power off

Use Case Diagram

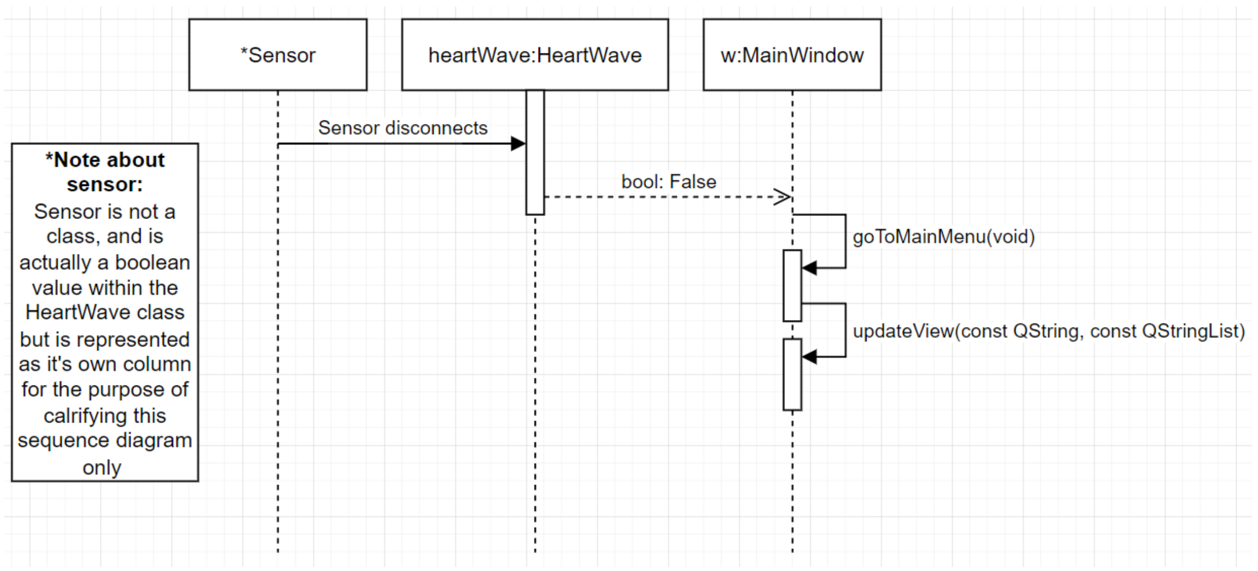

Use Case 2: Interruption Due to Sensor Off
<<include>>
Use Case 3: Battery Low
<<include>>
Use Case 1: Normal Operation
HeartWave
<<include>>
Use Case 4: Accessing a Session From Session Logs
<<include>>
User
Use Case 5: Changing Breath Pacer Settings
<<include>>
Use Case 6: Restoring Device to Initial Install Conditions
<<include>>
Use Case 7: Turn On Device
<<include>>
Use Case 8: Turn Off Device

Sequence Diagrams

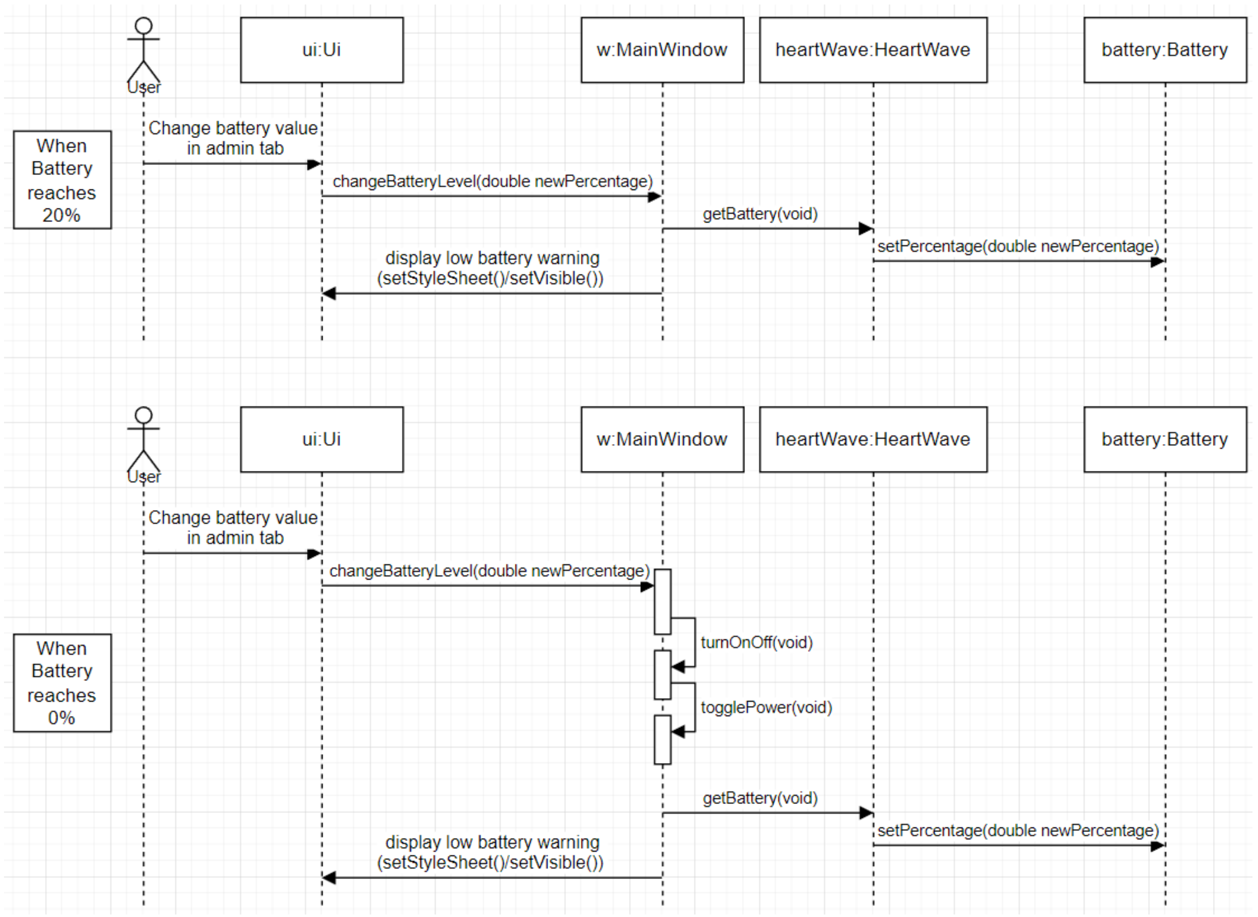
Sequence Diagram 1: Normal Operation



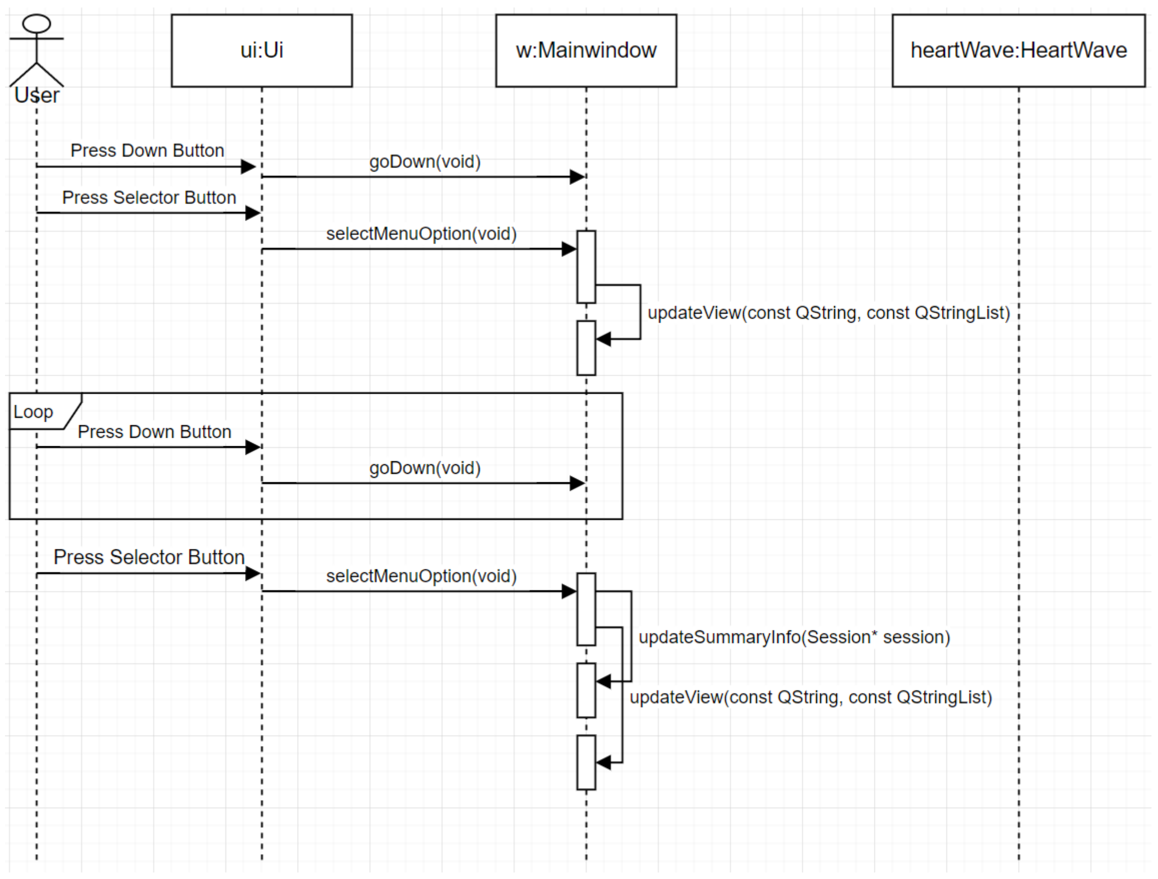
Sequence Diagram 2: Interruption Due to Sensor Off


*Sensor
heartWave:HeartWave
w:MainWindow
Sensor disconnects
bool: False
goToMainMenu(void)
updateView(const QString, const QStringList)
*Note about sensor:
Sensor is not a class, and is actually a boolean value within the HeartWave class but is represented as it's own column for the purpose of calrifying this sequence diagram only

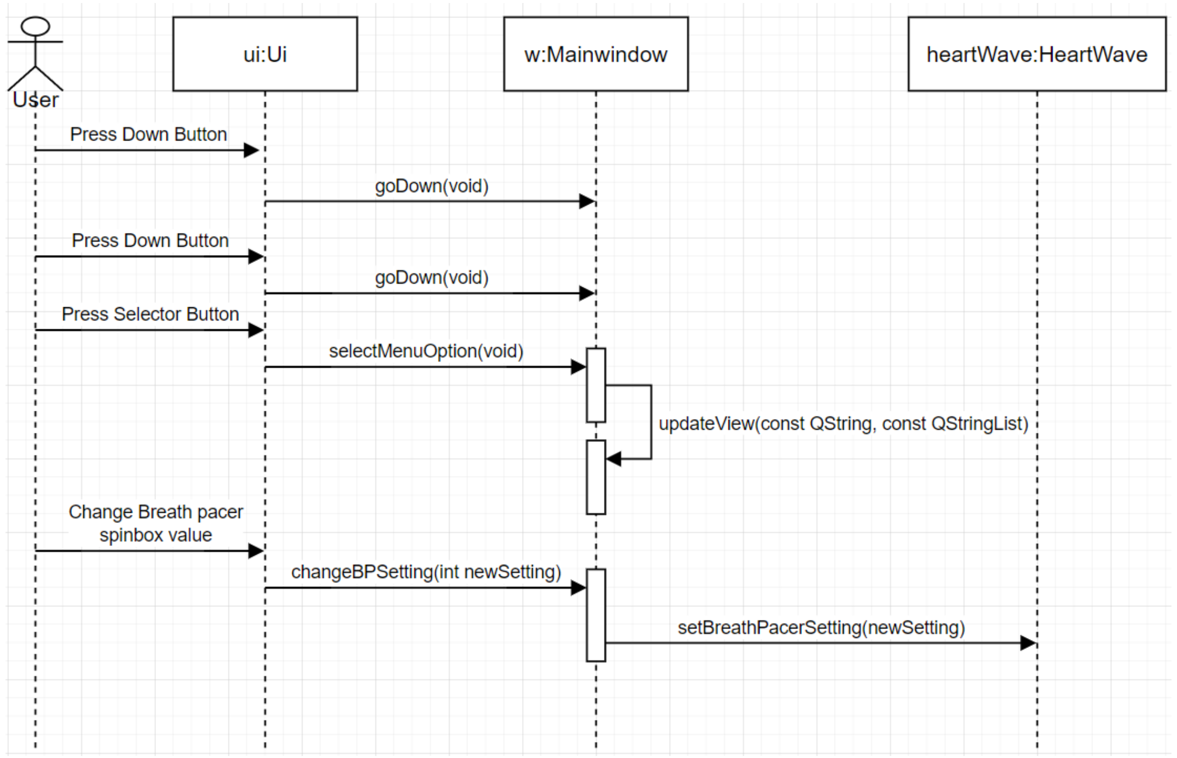
Sequence Diagram 3: Battery Low


User
ui:Ui
w:MainWindow
heartWave:HeartWave
battery:Battery
When Battery reaches 20%
Change battery value in admin tab
changeBatteryLevel(double newPercentage)
getBattery(void)
setPercentage(double newPercentage)
display low battery warning (setStyleSheet()/setVisible())
User
ui:Ui
w:MainWindow
heartWave:HeartWave
battery:Battery
When Battery reaches 0%
Change battery value in admin tab
changeBatteryLevel(double newPercentage)
turnOnOff(void)
togglePower(void)
getBattery(void)
setPercentage(double newPercentage)
display low battery warning (setStyleSheet()/setVisible())

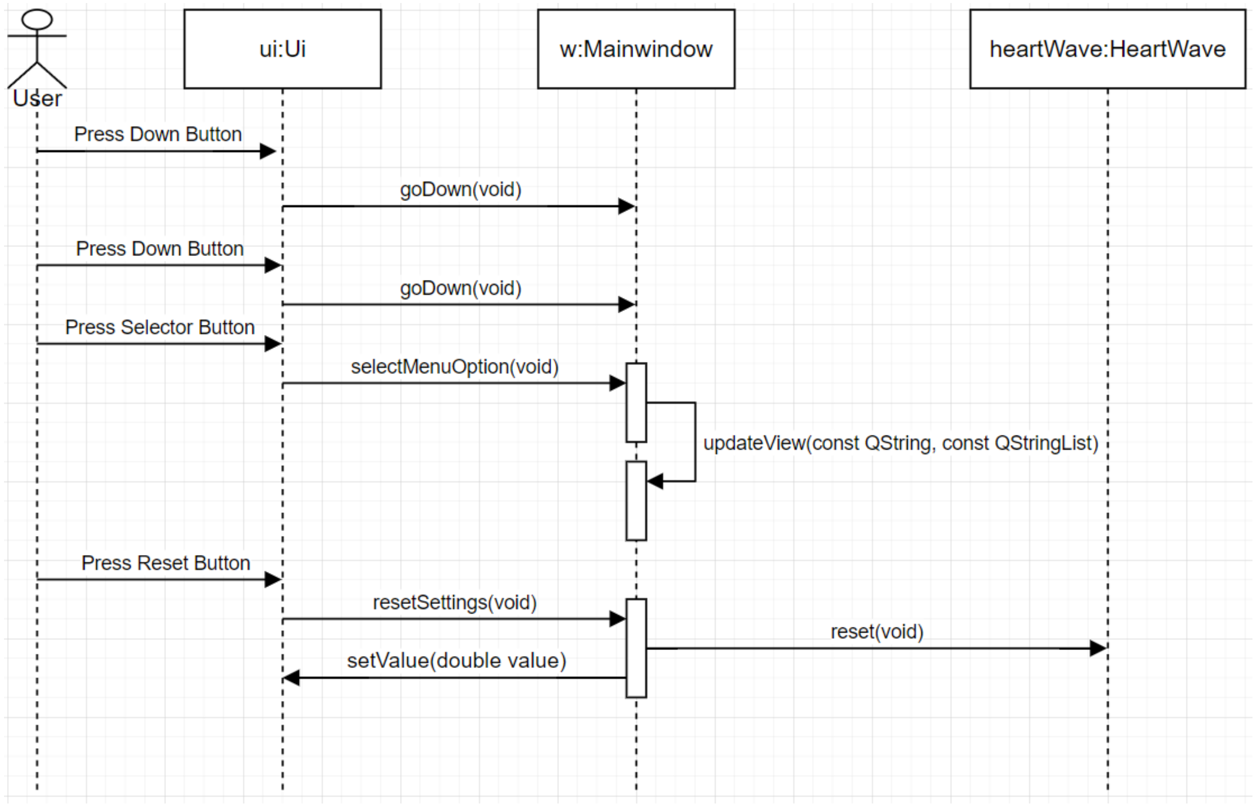
Sequence Diagram 4: Accessing a Session from Session Logs



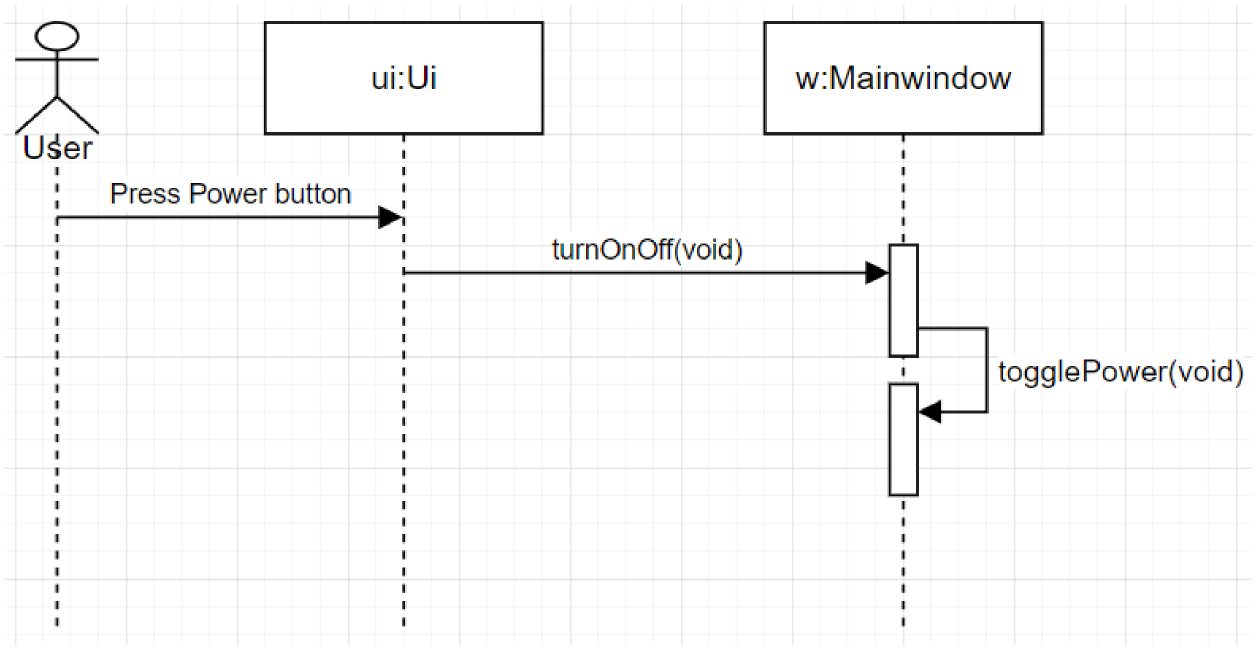
Sequence Diagram 5: Changing Breath Pacer Settings


User
ui:Ui
w:Mainwindow
heartWave:HeartWave
Press Down Button
goDown(void)
Press Down Button
goDown(void)
Press Selector Button
selectMenuOption(void)
updateView(const QString, const QStringList)
Change Breath pacer spinbox value
changeBPSetting(int newSetting)
setBreathPacerSetting(newSetting)

Sequence Diagram 6: Restoring Device to Initial Install Conditions



Sequence Diagram 7: Turn On/Off Device

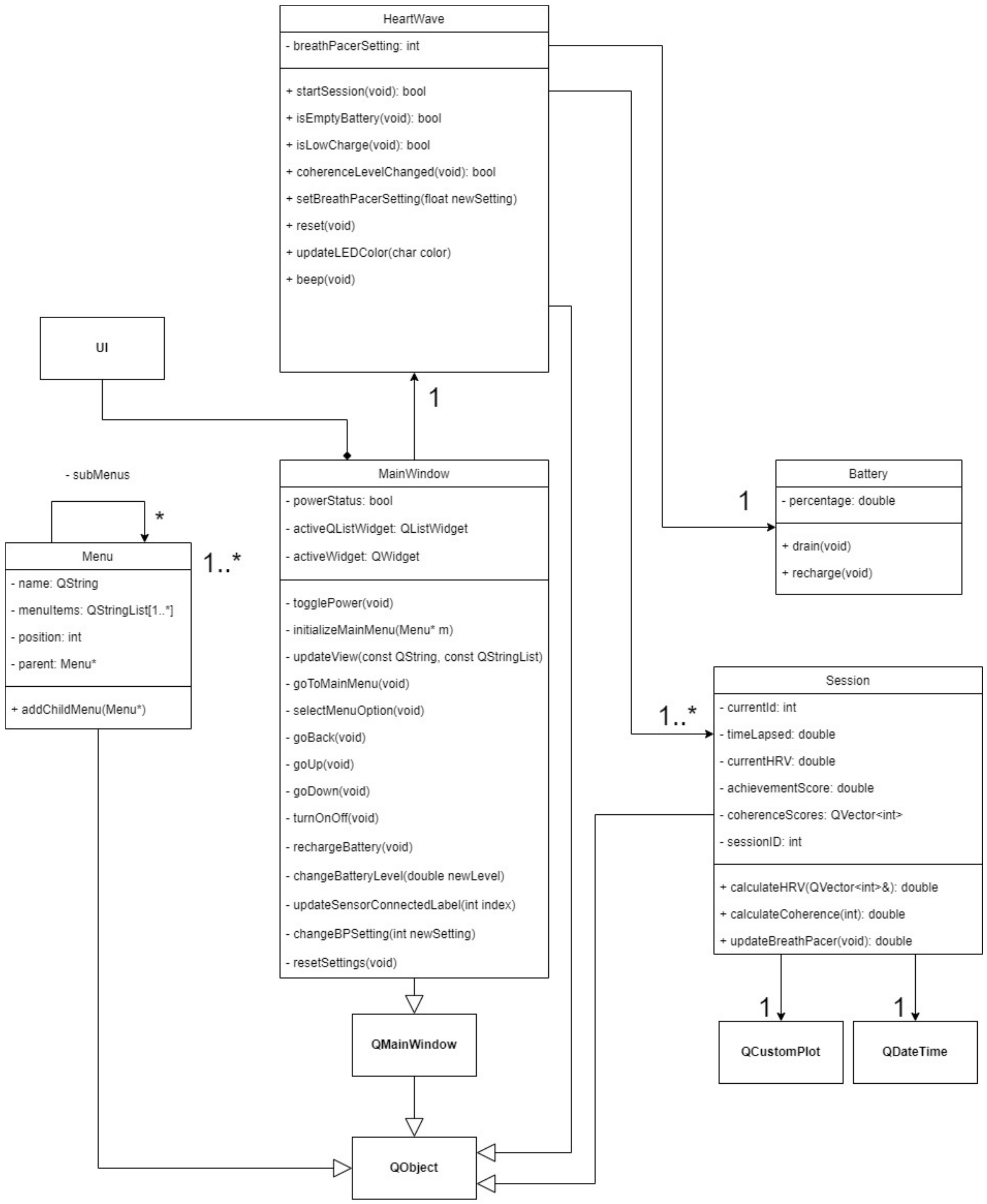


Traceability Matrix

ID	Requirement	Related Use Case	Fulfilled By	Test	Description
1	HeartWave is powered on by a user	Use Case 1: Normal Operation	Mainwindow.h Mainwindow.cpp HeartWave.h HeartWave.cpp Menu.h Menu.cpp Battery.h Battery.cpp Session.h Session.cpp Qcustomplot.h Qcustomplot.cpp mainwindow.ui	User turns on the device and is able to open the menu, adjust the battery, disconnect the sensor, and use the buttons to change the graph's coherence.	User presses the on button to initiate the session, the initial graph will appear and the user will be able to adjust the coherency, disconnect the sensor, deplete and recharge the battery, and access the menu.
2	Sensor is interrupted by the user/malfuncti ons	Use Case 2: Interruption due to sensor off.	Mainwindow.ui Mainwindow.h Mainwindow.cpp HeartWave.h HeartWave.cpp	Tested by toggling sensor connectivity in the Admin tab within the UI	If the sensor is disconnected/interrupted by the user, indicate that the pulse reading has been disabled and the session will then terminate.
3	The battery of the HeartWave device is <=20%.	Use Case 3: Battery low.	Mainwindow.ui Mainwindow.h Mainwindow.cpp Battery.h Battery.cpp	User drains the device battery to 20% and receives a warning, they can recharge or keep draining to 0% to shut off the program.	If the device is used until the battery reaches 20%, a warning will be issued to the user to indicate low battery and the need to recharge the device. If the user ignores this warning, the session will terminate at 0%.
4	User accesses a session from the session logs in the HeartWave device.	Use Case 4: Accessing a Session From Session Logs.	Mainwindow.ui Mainwindow.h Mainwindow.cpp Session.h Session.cpp Menu.h Menu.cpp	User clicks on the menu, goes to view session history, selects this option, and then selects which session to view.	The user is able to view previous session logs by clicking "view session history". They can use the arrow buttons to select the session they want to view and the information will display.

5	User opens the device menu to switch the settings of the HeartWave.	Use Case 5: Changing Breath Pacer Settings.	Mainwindow.ui Mainwindow.h Mainwindow.cpp Menu.h Menu.cpp	User goes to the menu and selects settings, the settings will be displayed and they can select the changes and apply them to the device.	User goes to the menu and selects settings, the settings will be displayed and they can select the changes and apply them to the device.
6	User restores the HeartWave to its initial install conditions.	Use Case 6: Restoring Device to Initial Install Conditions	Mainwindow.ui Mainwindow.h Mainwindow.cpp Menu.h Menu.cpp	User goes to the menu and selects “settings”. The settings are displayed and the user selects the reset button in the settings screen.	User goes to the menu and selects “settings”. The settings are displayed and the user selects the reset button in the settings screen. The HeartWave resets all settings to their default values successfully.
7	User turns on the HeartWave device.	Use Case 7: Turn On Device	Mainwindow.h Mainwindow.cpp HeartWave.h HeartWave.cpp	User clicks the power button while HeartWave is off.	User clicks the power button while HeartWave is off and the HeartWave powers on and the menu displays.
8	User turns off the HeartWave device.	Use Case 8: Turn Off Device	Mainwindow.h Mainwindow.cpp HeartWave.h HeartWave.cpp	User clicks the power button while HeartWave is on.	User clicks the power button while HeartWave is on and the HeartWave powers off and the display is powered off.

UML Diagram


HeartWave
- breathPacerSetting: int
+ startSession(void): bool
+ isEmptyBattery(void): bool
+ isLowCharge(void): bool
+ coherenceLevelChanged(void): bool
+ setBreathPacerSetting(float newSetting)
+ reset(void)
+ updateLEDColor(char color)
+ beep(void)
UI
1
- subMenus
*
Menu
1..*
- name: QString
- menuItems: QStringList[1..*]
- position: int
- parent: Menu*
+ addChildMenu(Menu*)
MainWindow
- powerStatus: bool
- activeQListWidget: QListWidget
- activeWidget: QWidget
- togglePower(void)
- initializeMainMenu(Menu* m)
- updateView(const QString, const QStringList)
- goToMainMenu(void)
- selectMenuOption(void)
- goBack(void)
- goUp(void)
- goDown(void)
- turnOnOff(void)
- rechargeBattery(void)
- changeBatteryLevel(double newLevel)
- updateSensorConnectedLabel(int index)
- changeBPSetting(int newSetting)
- resetSettings(void)
1
Battery
- percentage: double
+ drain(void)
+ recharge(void)
1..*
Session
- currentId: int
- timeLapsed: double
- currentHRV: double
- achievementScore: double
- coherenceScores: QVector<int>
- sessionID: int
+ calculateHRV(QVector<int>&): double
+ calculateCoherence(int): double
+ updateBreathPacer(void): double
1
1
QCustomPlot
QDateTime
QMainWindow
QObject

Textual Explanation

Design Overview
The program is divided into two main parts, classes responsible for updating and managing the UI (MainWindow, Menu), and classes responsible for the device's functionality (HeartWave, Battery, Session). The MainWindow class contains functions that update the UI and report any changes to the HeartWave class by calling the appropriate functions in the HeartWave class. It contains slots that upon triggering certain events, calls the corresponding function associated with that slot. The Menu class provides an interface for representing the UI views as c++ objects to allow for better manipulation/navigation between views. The HeartWave class acts as a kind of mediator between the MainWindow class and the other classes responsible for device functionality (Battery and Session). The HeartWave class is responsible for the device functionality, including starting a session, changing the breath pacer setting, resetting the device to initial install conditions, and many other functions. The Session class contains all information and functionality involved with sessions including calculating HRV, calculating coherence, and other such functionalities. The Battery class represents the device's battery, and contains all battery functionality.

Sequence Diagrams
For the sequence diagrams, we made three for each scenario specified in the assignment specification: normal operation, interruption due to sensor off, and battery low, in addition to sequence diagrams for the other added use cases. The sequence diagrams follow the sequence of function calls within the code and provide textual explanations when necessary in certain diagrams. Due to the design of the battery functionality, the battery does not deplete over time, but instead the user can change the value of the battery percentage from the admin tab within the UI. This is reflected within the Battery Low sequence diagram. In the sensor disconnected sequence diagram, the sensor is represented as its own class but in code it is represented by a boolean value within the HeartWave class, and is only represented as a class for the clarity and ease of representation within the diagram. The diagram also assumes that a session has already been started and the startSession function is running.

UML Diagram
For the UML Diagram, we made a UML class diagram containing all of the used classes. The structure of the program follows the mediator pattern somewhat as the HeartWave class acts as a mediator between the MainWindow class, which is responsible for updating and maintaining the UI, and the remaining classes responsible for various device functionality.

Traceability Matrix
For the Traceability Matrix, we included all 8 of our use cases along with an ID, description of requirements, the use case name, what .h/.ui/.cpp file fulfils it, how it can be tested, and a description of a successful run.

Unfulfilled Requirements

Due to time restrictions, the session functionality is incomplete, however, most of the functionality is implemented within the Session class, and what remains is the startSession() function in the HeartWave class. The session start is represented by a print statement in the console to indicate that a session has successfully been initiated.

The session history functionally is also part-way complete, due to the session functionality not fully working.