Traceability Matrix:

ID	Requirement	Related Use Case	Implemented By	Test	Description
1	The simulation must have an interactive interface, providing functional navigation buttons (selector, directional buttons, power button, menu button, charge button, as well as a return button) and a display.	N/A	mainwindow	Start the simulation to prompt the display and buttons, which can be clicked on for functionality.	Using the mainwindow.ui form, widgets representing the display and buttons are plotted and given functionality through the mainwindow class, which connects the widgets to the rest of the classes.
2	The simulation has a menu feature that is prompted by the menu button.	N/A	mainwindow, Menu	Once the simulation has been started, click on the menu button (the symbol has 3 lines) to open the menu and its options.	Upon starting the simulation, the menu will be initialized in mainwindow using an object from the menu class. By clicking the menu button, the class mainwindow will make the menu visible, providing options to the user. The Menu class has QStringList with a string for each menu option, which gets associated with various actions in mainwindow.
3	The simulation can be turned on/off using the power button.	N/A	mainwindow	Press the button with the power symbol at the bottom of the device. If the device is already on, it will turn off. Press it again to turn the simulation back on.	The widget representing the power button sets the display of the simulation to false, while also preventing other buttons from working.
4	The program allows the user to use the selector to start a session simulating heart rate variability readings	Heart Rate Variability Check (Use Case 1)	mainwindow, Menu, Device, Session	Once the menu has appeared, press the "New Session" option with the desired dataset option.	Once a new session starts, the mainwindow class will create a new session through its Menu object, which then initializes a session through the Device class, using data from one of three datasets. Periodic iterations are performed that update the display with new HRV and coherence data, handled by the Session pointer of Device, which is used to initialize data in a tuple.

5	The program must produce or imitate simulated heart rate readings and their subsequent coherence scores, updating every 5 seconds.	Heart Rate Variability Check (Use Case 1)	Session, Device	Either during or after a session, the HRV and coherence data shown through at the top of the screen, the summary and the graph all represent the simulated data	Upon initiating a session, the Session class will initialize a tuple with the relevant data that is displayed at the top of the screen. The Device class has a pointer of this session as well as an accompanying dataset for HRV and coherence scores.
6	The simulation has three lights that change colour during a session depending on the current coherence score of the user (red for low coherence, blue for medium coherence, green for high coherence).	Heart Rate Variability Check (Use Case 1)	mainwindow, Menu, Device, Session	Start a new session using one of the three datasets, during the simulation, look at the three coloured lights at the top of the screen and how they change in brightness depending on the coherence score	The widgets representing the 3 lights in mainwindow receive the coherence score values periodically from the Device pointer which retrieves information from the current session. The colour of each light will be changed to a lighter one, simulating the effects of it lighting up.
7	The information displayed during a session include the current coherence score, sum of coherence scores (achievement), the length of the current session as well as challenge level.	Heart Rate Variability Check (Use Case 1)	mainwindow, Menu, Device, Session	Start a new session, observe the text underneath the lights and their changing values. These represent the relevant data readings being simulated.	The text widgets in mainwindow responsible for displaying text are periodically updated with each iteration. Displaying the relevant information received from the Session tuple.
8	During the session, a graph plotting the heart rate variability over time is displayed to the user.	Heart Rate Variability Check (Use Case 1)	mainwindow, Menu, Device, Session	Start a new session using one of the three datasets, as the data is being retrieved the graph will update in real time with new HRV readings every 5 seconds.	Similar to the text display, the graph also receives periodic updates with values from the Session tuple, in particular the HRV and time values. The graph adds these coordinates to its plot as new information is sent.
9	A visual breath pacer must be displayed going through a forward/backward motion.	Heart Rate Variability Check (Use Case 1: extension 7ai)	mainwindow	When starting a new session, observe the bar beneath the lights. The bar will light up blue gradually in a back and forth motion.	mainwindow uses a bool breathVal to determine if the breath pacer bar is full or not. Using QTimer to gradually move the bar forward until it has reached the end, upon which it will return back at the same pace.
10	The breath pacer is preset to one breath per ten seconds, however can be changed in the menu options from 1-30 seconds.	Heart Rate Variability Check (Use Case 1)	mainwindow, Menu, Device, Settings	In the menu options, select the “Breath Pacer Interval” setting. The options through 1-30 will be available to change the timing of the	The Settings class has two member variables which are relevant to the breath pacer, two integers timeInterval and breath, which, depending on the selected option of the

				breath pacer.	user, will change from 1-30.
11	The user can end a session through the menu.	Heart Rate Variability Check (Use Case 1)	mainwindow, Menu	Once the session has stopped updating the graph and readings, the session can be ended by selecting the “End Session” option, which can only be selected after a session has been initiated.	Upon ending the session, the timers responsible for the breath pacer will be deinitialized. Furthermore, the Menu object itself will use its Device pointer to store the relevant information for session viewing.
12	Data pertaining to the session is shown at the end of the session, this includes the graph, average coherence score, achievement score, time spent in low, medium and high coherence scores (%), as well as length of the session	Heart Rate Variability Check (Use Case 1)	mainwindow, Menu, Device, Session	By ending the session, the summary data will be available by exiting the menu, where all the relevant data is presented. The graph will remain on the screen until a new session is started.	After the session has been ended, the current Session belonging to the Device pointer is accessed in the form of a tuple. Mainwindow will then use the data from the tuple to update the summary view values. The graph will remain the same, acting as a summary of the plotting from during the session.
13	A challenge level option is made available, allowing the users to change the ranges for which the coherence scores are considered low, medium and high.	Heart Rate Variability Check (Use Case 1)	mainwindow, Session, Settings	Press the menu button, the option “Challenge Level” will appear, which will supply 4 levels to choose from for future sessions.	Challenge settings are handled by the Settings pointer belonging to the Device. Once a user selects to change the challenge level through the interface, mainwindow initiates by altering the challenge level through the Device pointer in its Menu.
14	The simulation has the option to display previous sessions, with the accompanying summary data, as well as the exact time of the session.	Managing Session History (Use Case 2)	mainwindow, Menu, Device, Session, History	From the menu options, select “Show History”, which will display a selection of all previous sessions with their respective time stamps. Selecting one of these sessions will display the summary data as well as the HRV graph.	Once a session has been ended, the Session pointer will be added to a History pointer belonging to Device. Once the user chooses a session to view, a vector of all the sessions stored in History will be displayed by mainwindow. The session that the user then chooses will be loaded and have the same formatted summary as an end of session normally would. The graph will load the HRV dataset from the session being displayed,

					which acts as the current session.
15	The simulation provides the user the option to delete sessions from the system.	Managing Session History (Use Case 2)	mainwindow, Menu, Device, Session, History	From the menu options, selecting "Delete Session" will open a list of all previous sessions. Select the desired session for deletion. Alternatively, selecting "Delete All Sessions" will delete all the sessions at once.	The History class allows for sessions stored in its Session vector to be removed. By pressing the Delete Session option, the session pointer at the specified index will be erased, including all the relevant data stored inside of the session.
16	The simulation provides the option to perform a factory reset, clearing all the option settings and previous settings.	Reset Device (Use Case 3)	mainwindow, Menu, Device, History, Setting	Select the "Factory Reset" option from the menu, which will remove all session data and option preferences.	By accessing the Device pointer through mainwindow's Menu object, the "Factory Reset" resets the Setting pointer's member variables (challengeLevel and timeInterval) to their default. Selecting this option also clears the vector of Session pointers stored in the Device's History pointer.
17	The simulation must stop if there is an interruption in the signal of the simulation.	Heart Rate Variability Check (Use Case 1: extension 8b)	mainwindow, Menu, Device	While a session is taking place, click on the button with the heart icon. This will stop the simulation until the button is pressed again.	The heart button simulates the scenario in which the signal of the program is disrupted. When the button is pressed, the bool HeartContact is accessed from the Device through the Menu object in mainwindow, changing the boolean depending on if the button was already pressed before. Once the button is pressed, the flow of data through iterations will be stopped until the button is pressed again.
18	The simulation must have a battery charge display and the option to charge it through a button. If the battery shows 0%, then the simulation will not work until it is charged again.	Charge Device (Use Case 4)	mainwindow, Menu, Device	The battery percentage is visible in the upper left corner of the device screen, and will gradually decrease over time. Press the battery button to return the charge to 100%.	The charge button is connected to a Menu object through mainwindow. The Menu object contains a pointer of the Device class, which handles the battery. Upon every iteration of data, the battery depletes by 1 percent. If the battery reaches 0, then the program will not

					perform an iteration until it is recharged. Once the battery depletes, pressing the charge button will access the Device’s battery setter function and set it to a value of 100.