Traceability Matrix

ID	Requirement	Related Use Case	Implemented - by	Tested - by	Description
1	Battery Power depeletes when the application starts unless it is being charged.	Battery Power (UC17)	*MainWindow*, *RaDoTechDevice*, *Battery,* mainwindow.ui	Observe the battery progress bar in the ui to see the battery level decrease by 5% as the program runs, unless the charging button is pressed.	The Battery object within the RaDoTechDevice decreases by 20% every 15 seconds by starting the timer of the batterytimer QTimer object and stoping the timer of the chargedBatteryTimer Qtimer objec. This decrease is reflected in the battery progress bar in the ui. The process reverses, and the battery stops depleting when the charging button is pressed.
2	Battery Power charges properly.	Battery Power (UC17)	*MainWindow*, *RaDoTechDevice*, *Battery,* mainwindow.ui	Observe the battery progress bar in the ui to see the battery level increaseby 5% when the charging button is clicked, unless the button is pressed again.	When the charging button is pressed, the Battery object within the RaDoTechDevice increases by 5% every 15 seconds by starting chargedBatterytimer QTimer object and stoping the timer of the batteryTimer Qtimer object. This increase is reflected in the battery progress bar in the ui. The charging process reverses, and the battery stops charging when the button is pressed again.
3	Validate entered user data.	Create Profile (UC2)	*MainWindow*, mainwindow.ui	Displays a QMessageBox when the email or password fields in the "Create User" page are left empty.	The saveProfile function in *MainWindow* retrieves the email and password inputs, checks if they are empty, and halts profile creation if so. A QMessageBox notifies the user to fill in the required fields.
4	Validate entered profile data.	Create Profile (UC 2)	MainWindow, mainwindow.ui	Displays a QMessageBox when any required field (name, gender, weight, or height) in the "Create Profile" page is left empty.	The saveProfile function in *MainWindow* retrieves the name, gender, weight, and height inputs, checks for empty fields, and halts profile creation if any are empty. A QMessageBox notifies the user to complete all required fields.
5	Profile created ad associateed with	Create Profile	*MainWindow*, mainwindow.ui,	The creation of a new profile triggers	A new *Profile* object is initialized with provided data, added to the

	the user.	(UC2) & Add Multiple Profiles (UC5)	*Profile*, *User*	the addition of the profile to the currUser object. The updated profile list length is printed to the terminal, and the new profile is displayed on the profiles page of the app UI.	currUser's profile list, and displayed in the app's profile page UI.
6	Validate entered created User data upon login.	Login to App (UC6)	*MainWindow*, mainwindow.ui, *User*	Displays a QMessageBox warning when the entered email and password do not match the stored user credentials.	handleLogin function in *MainWindow* validates login credentials by comparing email and password inputs with those stored in the *User* object and appropriately triggers a QMessageBox.
7	Scanning when only paired with the app.	Start Health Scan (UC7)	*RaDoTechDevice, MainWindow,* mainwindow.ui	A QMessageBox notifies the user to pair the device with the app before initiating a scan.	The isPaired flag in RaDoTechDevice defaults to false. Scans proceed only when the flag is set to true via the 'Pair to app' toggle. Otherwise, a warning QMessageBox is displayed.
8	Not able to turn the device on when battery power completely diminishes.	N/A	*MainWindow,* mainwindow.ui	Both turn off and turn on buttons disable when battery reach to 0%.	If the MainWindow tries to get the value of the RaDoTechDevice object's battery value when it reaches to 0 it will disable the Turn on button on the ui's interface.
9	Battery depletion stops when the device turns off.	N/A	*MainWindow,* mainwindow.ui	The batteryPowerProgr-essBar halts decrementing every 15 seconds when the "Turn Off" button is clicked, confirming the device is off.	The batteryTimer (a QTimer object in MainWindow) stops sending timeout signals upon receiving the "Turn Off" button click event.
10	Inform user of low battery.	Battery Power (UC17)	*MainWindow,* mainwindow.ui	When battery power drops to 20%, a QMessageBox is triggered to alert the user of low.	The *MainWindow* calls the isLowBattery function of the *RaDoTechDevice* object, which accesses the *Battery* object to check if its value is 20 or below. If true, the function returns a

				battery.	boolean true, prompting the MainWindow to display a QMessageBox warning the user of low battery on the UI.
11	Selecting a profile before measuring health scans.	Start Health Scan (UC7)	*MainWindow,* mainwindow.ui	A QMessageBox will be displayed when user selects “scan” on the device before selecting “Measure now” on the app.	Assigning an already existing Profile object to the currProfile pointer in the MainWindow before performing scans for it.
12	Measuring each Health Scan.	Data processing (UC10)	*MainWindow*, mainwindow.ui, *DataProcessor*	Observing that each scan is completed via a message saying so via the QLabel below the QPushButton of the scan button in the device view.	everytime the device receives one user scan by pressing the “scan” button the user is instructed to go to the app’s side to press the “next” button to measure the scan point.
13	Select a specific Historical data when trying to Viewing one of them.	View Historical data (UC11)	*MainWindow*, mainwindow.ui	A QMessageBox will be displayed if the user has not selected a record.	If the currentItem pointer in the historical data list QListWidget points to a nullptr then the Mentioned QMessageBox will be displayed.
14	Viewing Historical data info.	View Historical data (UC11)	*MainWindow,* mainwindow.ui, *HealthData, visualization*	Observing the ui showing indicator, visualization, comments and recommendations of selected historical data after clicking the “view details” button in the history page.	Checks for each healthl data in the stored for profile until a historical data date matches up with the selected one and updates the ui according to the indicator, comments visual graphs and recommendations of the historical data.
15	Making sure the the chart graph in historical data is accurate.	View Chart Visualization from Metrics (UC13)	*MainWindow,* mainwindow.ui, *visualization*	Printing out the value for each conductance value in the healthData and seeing if the values on the chart graph in the visualization tab of the metering result page in the ui	Gets the conductance values from the passed in pointer of the HealthData object into the showBarGraph func which then displays a bar chart of the HealthData values onto the ui visualizing said data into a bar chart for the user.

				match with those printed in the terminal.	
16	Making sure the the circle graph in historical data is accurate.	View Circle Visualization from Metrics (UC14)	*MainWindow,* mainwindow.ui, *visualization*	Printing out the value for each conductance value in the healthData and seeing if the values on the circle graph in the visualization tab of the metering result page in the ui match with those printed in the terminal.	Gets the conductance values from the passed in pointer of the HealthData object into the showCircleGraph func which then displays a circle graph of the HealthData values onto the ui visualizing said data into a circle graph for the user.
17	Making sure the the body graph in historical data is accurate.	View Body Visualization from Metrics (UC12)	*MainWindow,* mainwindow.ui, *visualization*	Printing out the value for each conductance value in the healthData and seeing if the values on the body graph in the visualization tab of the metering result page in the ui match with those printed in the terminal.	Gets the conductance values from the passed in pointer of the HealthData object into the showBodyGraph func which then displays a body graph of the HealthData values onto the ui visualizing said data into a body graph for the user.
18	Making sure the indicators in historical data is accurate.	N/A	*MainWindow,* mainwindow.ui, *HealthData*	Clicking on the metering result's indicator tab to see if the Health data algorithms match with the values displayed in the history page of the app view of the ui.	There are algorithms in HealthData that calculate all of the values displayed in the indicator tab of the historical data.
19	Health data's Comments in the ui are accurate.	N/A	*MainWindow,* mainwindow.ui, *HealthData*	All the Profile's comments are displayed in the comments tab of the metering results in the history page of the app view of the ui.	The comments the user entered are all stored into the HealthData object created when all the scans are done which are used to access them and display them in the ui.

20	Health data's recommendation's in the ui are accurate.	Recommend Medical Advice from Metrics (UC15)	*MainWindow,* mainwindow.ui	Observing recommendation placeholder is displayed in the recommendation's tab of the metering results in the history page of the app view.	Each Historical data has a hardcoded placeholder for recommendations.
21	Deleting a profile.	Delete Profile (UC4)	*MainWindow,* mainwindow.ui, *Profile*	Observing if the profile has truely been deleted by visiting the profiles page in the app view of the ui.	Traverse the profiles list in the currUser object to find a matching profile to the user selected one and free its pointer from the heap.
22	The application does not contain any memory leaks.	N/A	N/A	Run valgrind to check for memory leaks.	All dynamically allocated memory that the program was designed to allocate is deleted in the appropriate class destructor.
23	Edit already existing profile data.	Update Profile (UC3)	*MainWindow,* mainwindow.ui, *Profile*	Viewing profile data after editing it and observing the changes.	Setting the new values that were typed into the profile data fields to the profile the currProfile pointer points to when the user presses the save button on the edit profile page of the ui.
24	Add Multiple Profiles.	Adding Multiple Profiles (UC5)	*MainWindow, Profile, User, mainwindow.ui*	View Profiles in the profiles tab of the app view after add a profile	When creating a profile via clicking a create profile button takes the user to a create profile page which upon being saved is added to the profiles list of the User object.
25	Scannig must be on skin contact for at least 5 seconds to proceed to the next scan.	Perform Scan (UC8)	*MainWindow, mainwindow.ui*	If User clicks take off skin button too early a QMessageBox displays informing the User that they need to redo the scan because the previous one was invalid.	A QTimer with the interval of 5 seconds start as soon as the User clicks the scan button and the User will need to click on the button once again right after the timer hits the 5 second mark to take it off their skin and complete the scan