Use Case 1: Add User
Primary Actor: User
Stakeholders and Interests:
User: - User Wants to be able to add a User in the App.
Precondition: None
Success Guarantee: The user adds another user to the app, using the app.
Main Success Scenario:

1. User opens the app
2. User navigates to the “Profiles” tab,
    1. User selects the button labeled “New Profile”
3. New Page is Open with text boxes labeled
    1. Weight, Height, Name,
    2. Gender CheckBox
4. User the button labeled “Save”
    1. User is added to the list of users.
    2. User is returned to “Profile Info” tab

Extensions:

<All steps> Application crashes.

a1. App crashes

a2. Information is not saved and the user restarts at Step 1.

3.1 User did not properly fill in the “Height” or “Weight” textbook

3.2 The application does not allow you to save.

Use Case 2: Change Current User
Primary Actor: User
Stakeholders and Interests:
User: - User changes which user is Active
Precondition: At least two profiles must exist
Success Guarantee: The user Changes who is active in the app, using the app.
Main Success Scenario:

1. User opens the app
2. User navigates to the “Profiles” tab,
    1. All Users Profiles are displayed.
    2. User, clicks on a Profile “Name”
3. User selects the Check-Box labeled “Profile for metering”
    1. User clicks “save” or “back.png”
4. User is returned to the “Profiles” tab
    1. The “current.png”, is placed on the Profile that was selected.

Extensions:

<All steps> Application crashes.

a1. App crashes

a2. Information is not saved and the user restarts at Step 1

Use Case 3: Deleting User
Primary Actor: User
Stakeholders and Interests:
User: - User wants to be able to delete(s) a user's Profile, using the app.
Precondition: A profile must exist
Success Guarantee: The user deletes another user from the app.
Main Success Scenario:

1. User opens the app
2. User navigates to the “Profiles” tab
    1. All User(s) Profile(s) are displayed.
3. User Clicks on a Profile
4. User Clicks on “Delete Profile” Button
    1. User is returned to the “Profiles” tab, the selected Profile is absent

Extensions:

<All steps> Application crashes.

a1. App crashes

a2. Profile is not deleted and the user restarts at Step 1.

Use Case 4: Updating User Profile
Primary Actor: User
Stakeholders and Interests:
User: - User Wants to be able to update an existing profile.
Precondition: None
Success Guarantee: The user successfully updates a pre-existing profile, using the app.
Main Success Scenario:

1. User opens the app
2. User navigates to the “Profiles” tab
    1. A dropdown with all User(s) Profile(s) are displayed.
3. User clicks on a Profile, in the dropdown tab
    1. The Profile information is displayed.
4. User Updates the Name, Weight or Height textboxes.
5. User select the button labeled “save”
    1. App is updated with the new information
6. User is returned to the “Profile Info” tab

Extensions:

<All steps> Application crashes.

a1. App crashes

a2. Information is not updated and the user restarts at Step 1.

<4a> The updated information is empty

<4a.1> The app does not allow the user the current changes.

Use Case 5: Using Radotech

Primary Actor: User

Stakeholders and Interests:

User: - Use updates the current profile with Radotech readings.

Precondition: Radotech device must be on, and a profile must be selected

Success Guarantee: Users profile is successfully updated with Radotech readings.

Main Success Scenario:

1. User Opens App
2. User navigates to the “Measure” tab
3. User Clicks “Measure Now”
4. User places sensor in the first 1 of 24 positions
    1. Reading was measured as the Graph is updated.
    2. User Clicks on the button next to “Now Measuring: “
5. Go through all 24 points.
6. App is updated with all the sensor readings to the current profile.

Extensions:

<All steps> Application crashes.

a1. App crashes

a2. Readings are not saved, User restarts at step 1

2a: Radotech is Off, user is prompted by a Message Box

2a.1: User turns on “Radotech”

2a.2: User continues to step 3

4a. User Turns off Radotech during “Reading”

4a.1 The readings that were displayed are not saved

4a.2 User Restarts at Step 3.

4b. Radotech battery is at 0%,

4b.1 User is prompted by a message box

4b.2 User charges battery,

4b.3 Readings that were displayed are not saved

4b.3 User restarts at Step 3.

4c. User simulates all readings at once

4c.1 User selects the simulate readings checkbox

4c.2 App generates readings

4c.3 User is placed in Step 6

Use Case 6: Display Reading
Primary Actor: User
Stakeholders and Interests:
User: - User wishes to view the readings
Precondition: A reading must exist
Success Guarantee: User has viewed the readings.
Main Success Scenario:

1. User Opens App
2. User navigates to the “History” tab
    1. User Select a reading
3. Bar Graph is displayed
    1. User views the readings.
    2. User switched to other chart views if needed.

Extensions:

<All steps> Application crashes.

a1. App crashes

a2. User restarts at step 1

Use Case 7: Recommendation Tab
Primary Actor: User
Stakeholders and Interests:
User: - User wishes to get recommendation and the Doctor/Specialist recommendation
Precondition: None
Success Guarantee: Users have viewed the Recommendation
Main Success Scenario:

1. User Opens App
2. User navigates to the “Recommendation” tab
3. App, generates recommendation based on the User, poor readings
    1. User inputs the Doctor/Specialist recommendation into the textbox
    2. User Clicks Save
4. User Selects the “Back.png” button

Optional:

3a: User wishes to clear recommendation textbox

3a.1: User Selects “cancel” button

3a.2: User Selects “Save” Button

3a.3 The textbook is updated as blank

Extensions:

<All steps> Application crashes.

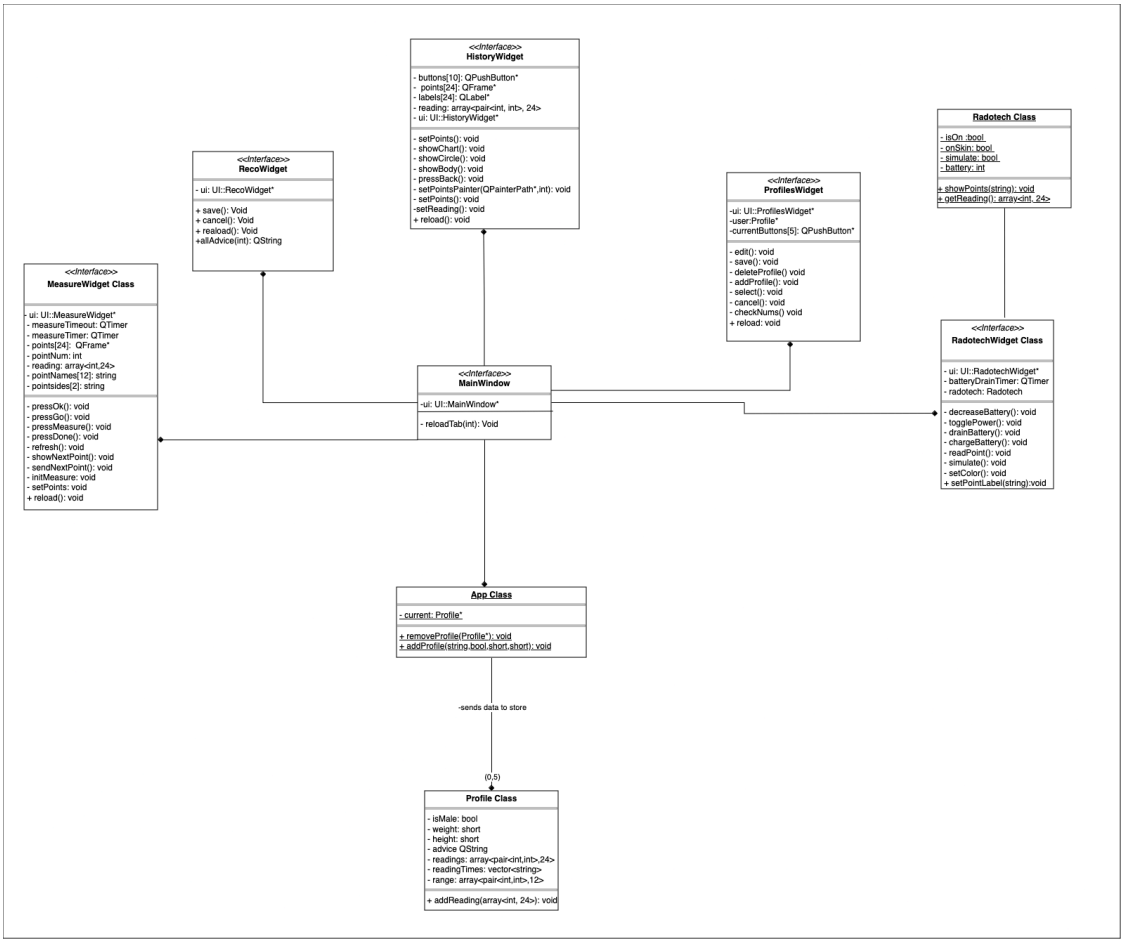
a1. App crashes

a2. User restarts at step 1

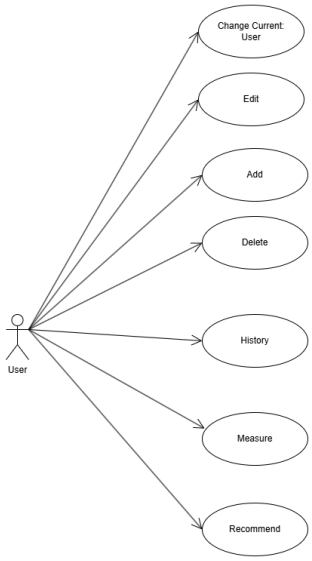
3a. User does not Click “Save” button

3a1. The recommendation text in the textbox is not saved

UML Class Diagram

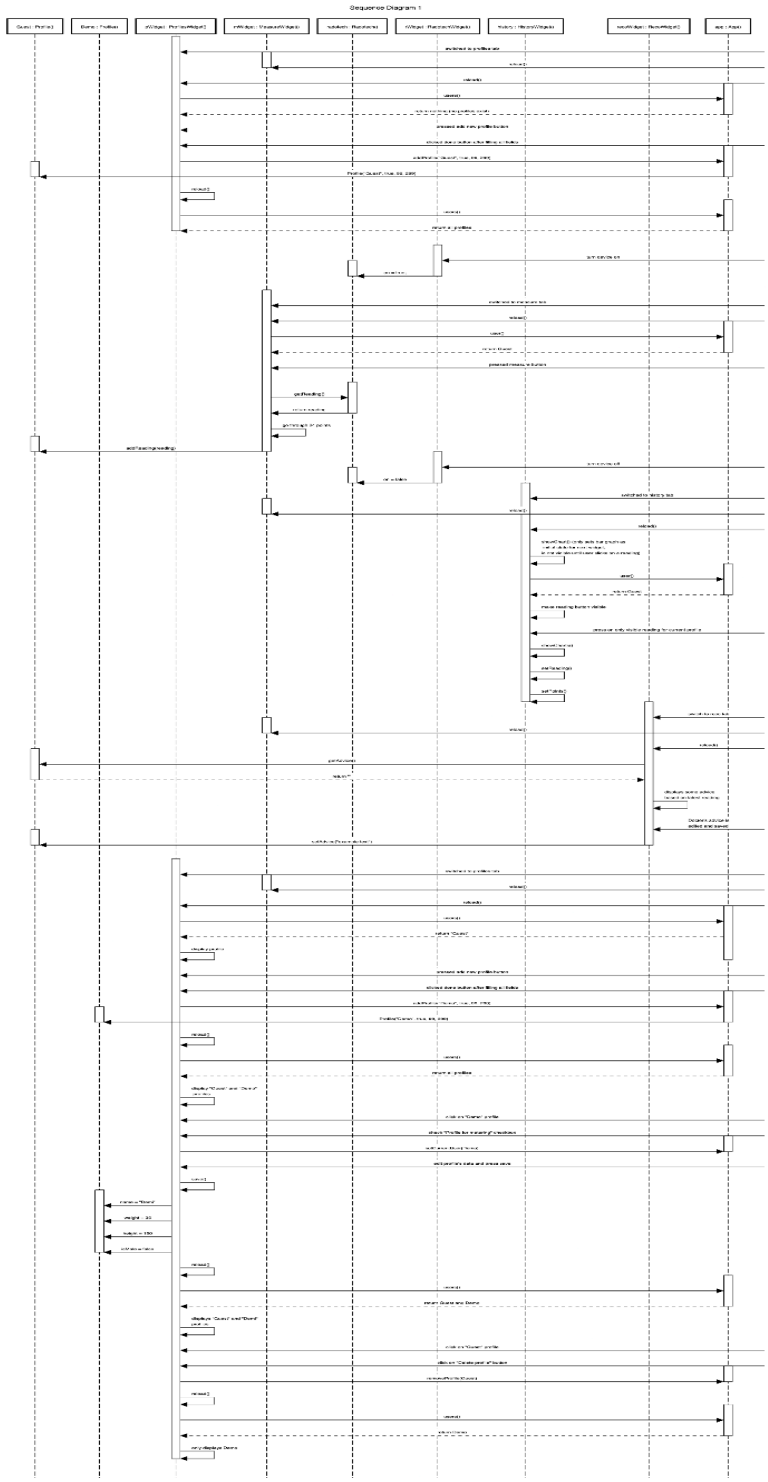


Use Case Diagram



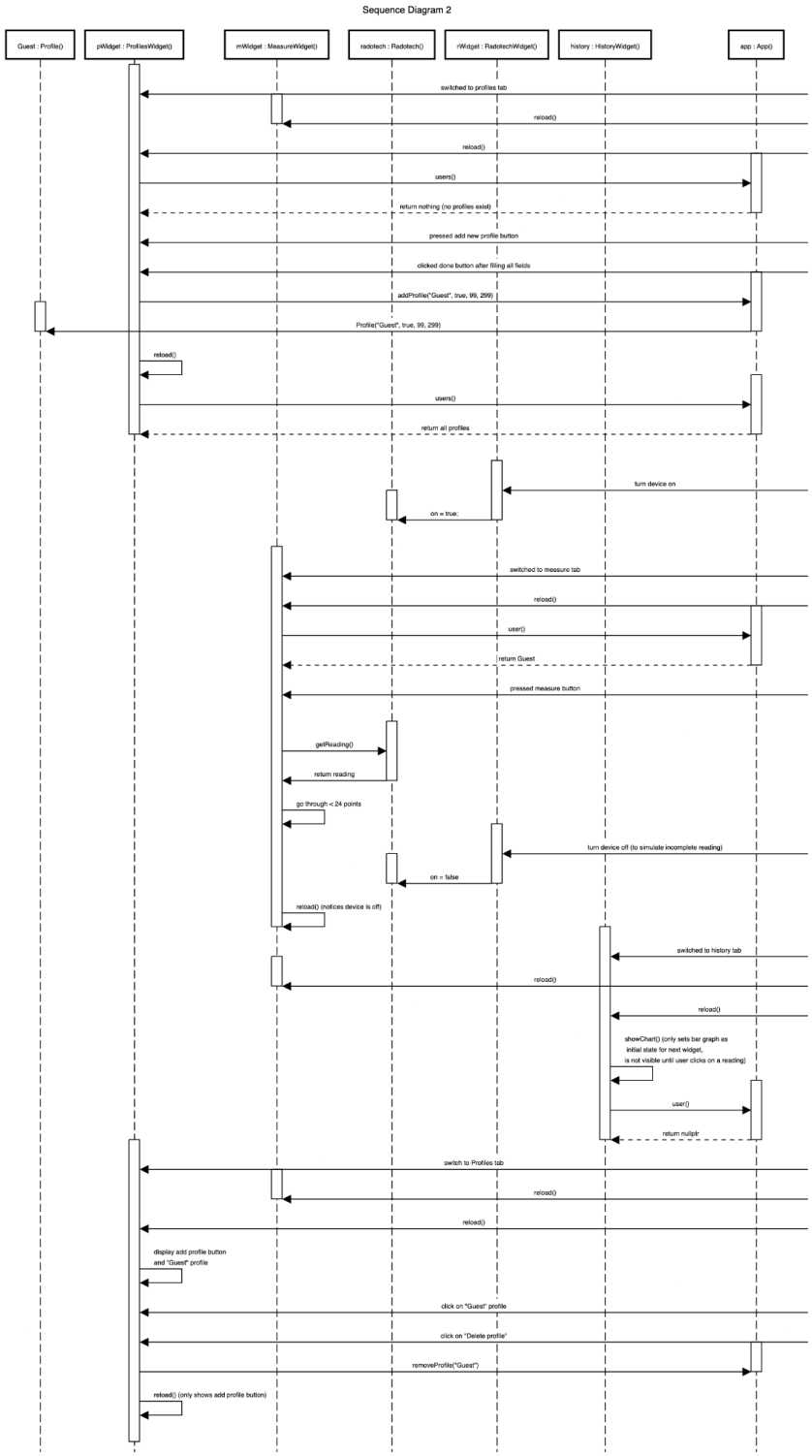
Regular Operation Sequence Diagram

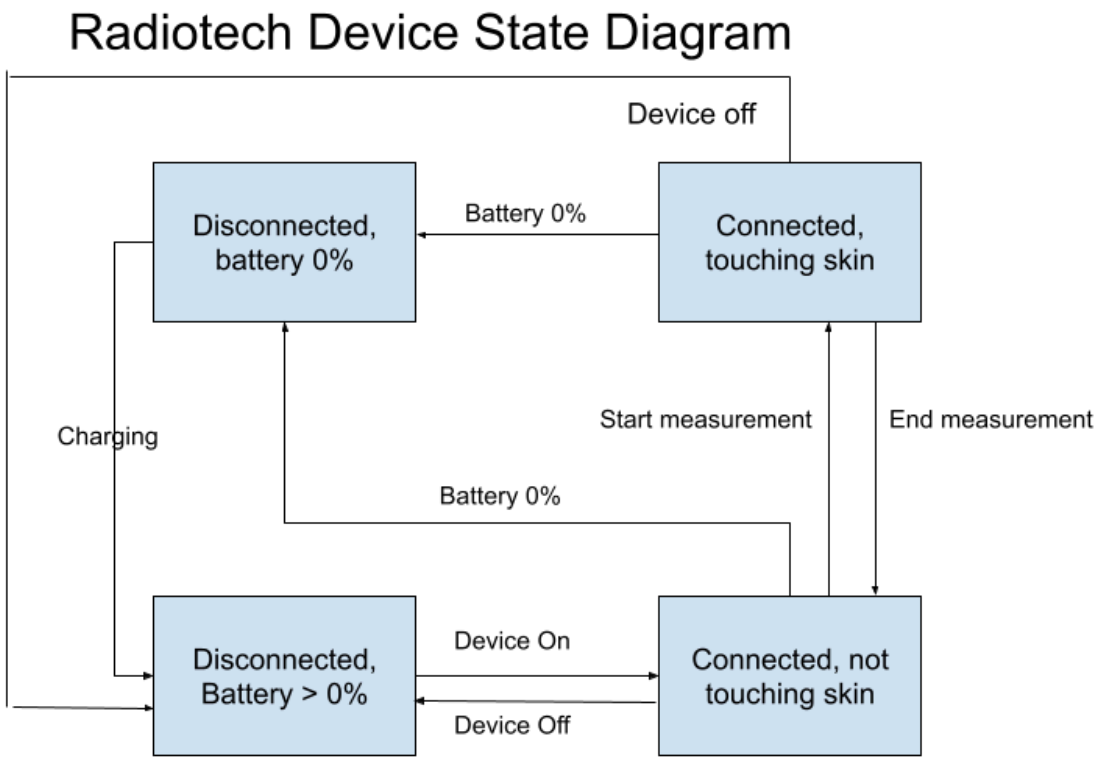
The diagram goes through the creation of a user and taking a successful reading for the user and viewing the results of the graph in the History tab. The diagram also goes through the creation of an additional profile, selecting the new profile as the “Profile for metering”, updating user info, and the deletion of the first profile.



Safety Operation Sequence Diagram

This diagram goes over what happens in the case of an incomplete reading due to loss of connection or the device turning off. A profile is first created just as above, but during the reading the Radotech device turns off, which prevents the saving of the reading, to the current user. The sequence diagram ends with the deletion of the only profile.

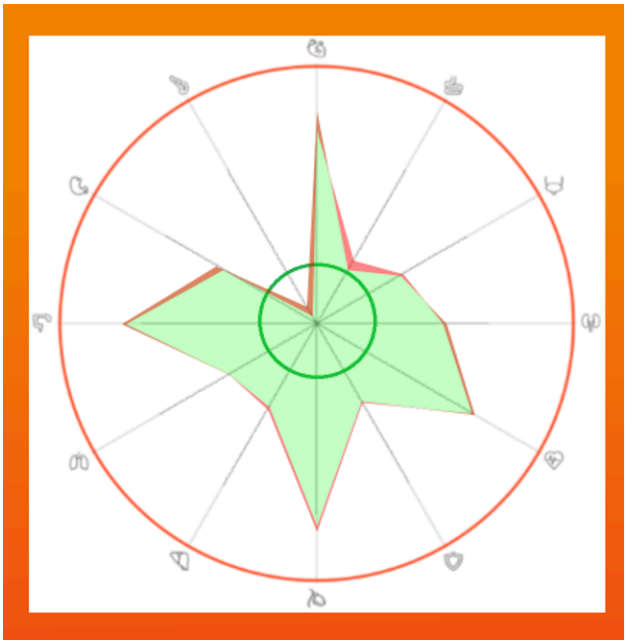




Design Explanation

Since the Radotech device would not record a reading if the device were not on the skin, our implementation assumes that the user has placed the sensor on their skin as soon as they click on the button that displays the point on their hand/foot they are reading. This is implicit and can only be confirmed when the user is in the Measure tab and is waiting for a point to show up on the reading graph. Alternatively, the person running the simulator can check the simulate checkbox, this option does not reflect how the device actually works and is included for convenience.

Circle chart has a minimum reading amount, this prevents you from getting a jagged and off looking graph. The Radotech app also has this implementation, we believe it is important to reiterate our own understanding of why we manipulate the readings. You could argue it is better to get the raw data then, get a manipulated graph but you can see the raw numbers in the “Chart” tab in the “History” Tab. The point of the Circle tab is to easily compare your left and right-side reading. Here is an example to show.



We make use of the Singleton, Mediator, Strategy, and Observer design patterns. The App and Radotech classes are singletons, where the App class is only instantiated in the MainWindow ui class, and Radotech has its only instance declared in RadotechWidget. Any interaction with these two from other widgets is done by using static members and methods. This is suitable since it allows every widget to access the same set of profiles through the App class, and the MeasureWidget does not need its own instance of the Radotech class, instead it interacts with the only instance of Radotech using static methods.

The Radotech class also acts as a Mediator between the MeasureWidget and RadotechWidget interfaces, the button that displays what point a user is measuring during a reading is passed to the RadotechWidget through the Radotech class, and this function call originates in the MeasureWidget class. The RadotechWidget makes use of the Strategy behavioural design pattern by offering the user two different ways to simulate a skin-on scenario, either by clicking on the button with the point name, or by checking the simulate checkbox.

Finally, the MeasureWidget makes use of the Observer behavioural design pattern by making use of a QTimer to check if the Radotech device is still on. The QTimer checks with the Radotech class by calling a static method that tells the widget if the device is still on, the timer checks this every 100 milliseconds. This timer starts when the measure button is pressed (i.e. the reading has started), and keeps going until the user has measured all 24 points. Other than this, we do not use the Observer pattern since all the other widgets do not have to worry about the state of another widget changing.

Requirements Traceability Matrix

ID	Requirement	Relates Use Case(s)	Fulfilled By	Tested by	Explanation
1	Create, update, and delete user profiles. Each device should support multiple profiles (up to five).	UC1: Add User UC2: Change Current User UC3: Deleting User UC4: Updating user profile	Profile.h Profile.cpp profileswidget.h profileswidget.cpp profileswidget.ui App.h App.cpp	Creating new users, updating them, and deleting them. Testing out edge cases by going all the way to 5 and down to 0.	The Profiles tab displays all registered profiles and has a button for adding a new profile. Clicking on a profile allows it to be edited, deleted, or set as the active profile.
2	Interface with RaDoTech hardware to collect health data.	UC5: Using RaDoTech	Radotech.h Radotech.cpp radotechwidget.h radotechwidget.cpp radotechwidget.ui	Taking a measurement after turning on Radotech and checking in History.	The RaDoTech device is simulated in the Radotech and radotechwidget classes.
3	Show the device contacting or not contacting the skin. Successive measurements require lifting the device off the skin and putting it back on the next point.	UC5: Using RaDoTech	Radotech.h Radotech.cpp radotechwidget.h radotechwidget.cpp radotechwidget.ui	Ensure that the next point only shows up if the simulate checkbox has been checked, or the button has been pressed.	The process of moving the device and putting it on the skin is simulated in the app, each measurement only requires pushing a button.
4	Process raw data to generate health metrics as you understand it based on 1) answers you received from the Product Owner, and/or 2) from your own research.	UC5: Using RaDoTech	Profile.h Profile.cpp measurewidget.h measurewidget.cpp measurewidget.ui	Ensure that data visualized in the History tab matches data generated during a reading.	Once a reading is completed, each value is marked as normal, below normal, or above normal. This is used to generate the specialist recommendation.
5	Display health metrics in an easy-to-understand, visual format within the application.	UC6: Display Reading	measurewidget.h measurewidget.cpp measurewidget.ui historywidget.h historywidget.cpp historywidget.ui	See video demonstration.	Readings are plotted on a graph while measurements are being taken. Once the measurement

					process is complete, the readings can be displayed as a bar graph, circle graph, or body chart.
6	Provide a place-holder for specialists' recommendation.	UC7: Recommendation Tab	recowidget.h recowidget.cpp recowidget.ui	Recommendations are stored for users. Ensure no errors when users are deleted/added.	The app analyzes the most recent reading and will display advice for any measurements that are above or below normal.
7	Store and allow users to access historical health data for trend analysis.	UC6: Display Reading	historywidget.h historywidget.cpp historywidget.ui	See video demonstration.	The history tab has a list of the 10 most recent readings, each one can be displayed as a bar graph, circle graph, or body chart.
8	Show charge depletion and low power indication.	UC5: Using RaDoTech	radotechwidget.h radotechwidget.cpp radotechwidget.ui	Ensure battery level decreases every 2 seconds, and LED on device turns red for low power indication.	There is a battery indicator with the battery percentage displayed next to the RaDoTech device.