ElderlyTracker - 325 A3 Report

Gitlab

Link to GitLab repository: https://gitlab.ecs.vuw.ac.nz/browntoma/Swen325Ass3

UX Decisions

Battery Status Pie Charts

Having a list of pie charts to show the status of each battery is one of the UX decisions I have made in my application. This decision I made was based off real life experiences I have faced while using electric appliances. I had drafted up two options, one displaying the battery percentages as donut graphs and one as numerical values. Both designs are shown in the images below.

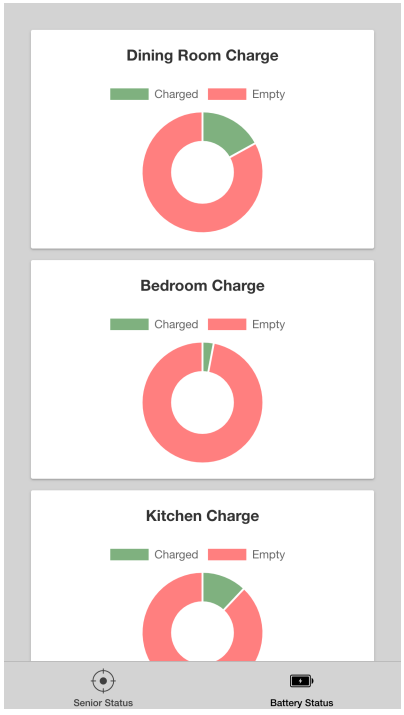


Image 1.1

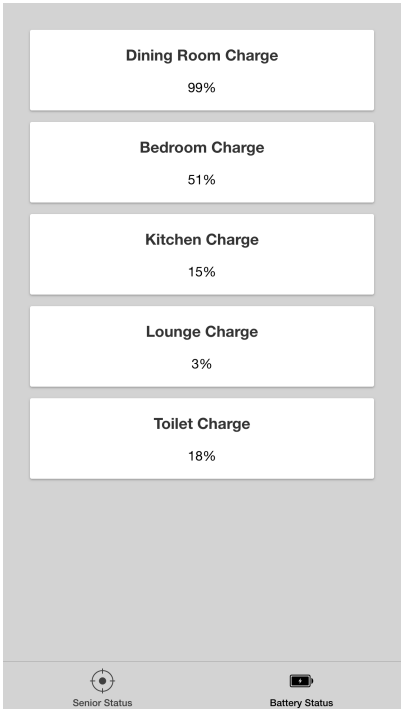


Image 1.2

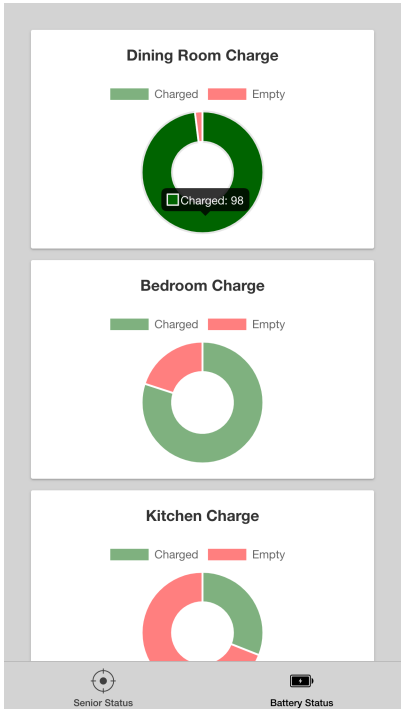
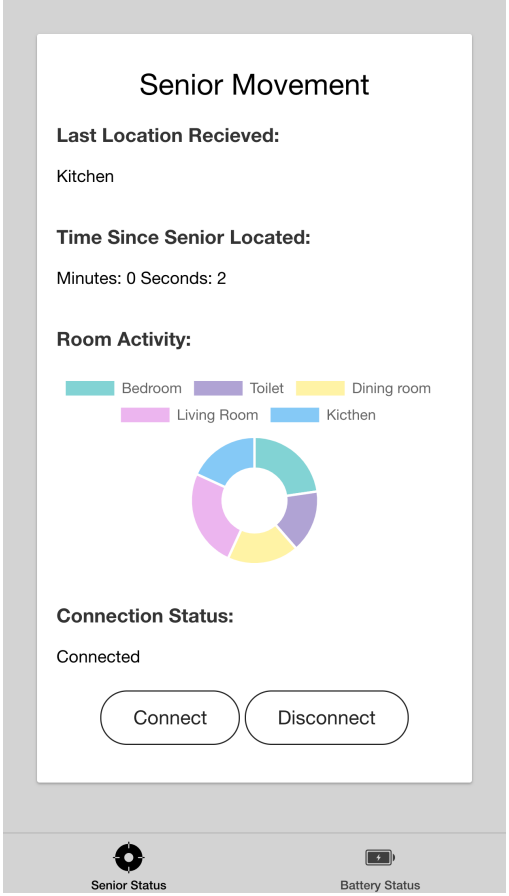


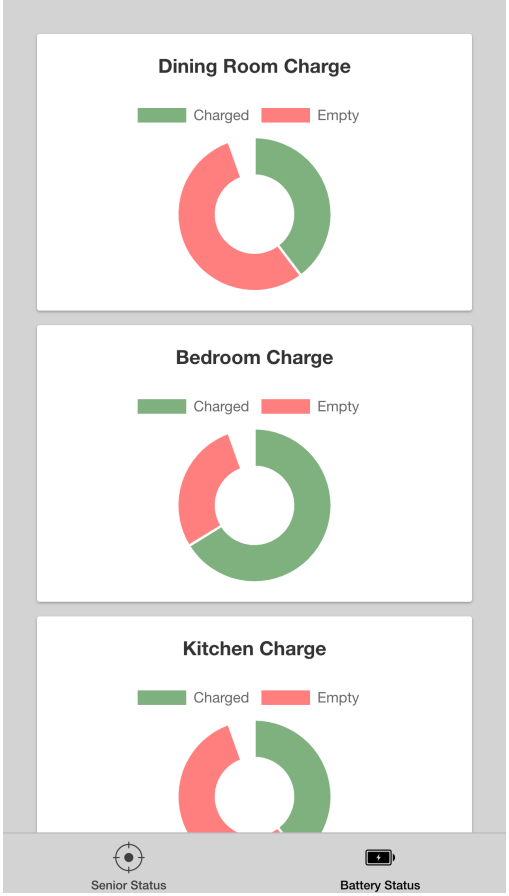
Image 1.3

I decided to use the pie charts as it shows the user a very quick representation of the battery of each sensor. The user does not have to read any numerical values to get an idea of the percentage and can just see how much 'green' is in the chart. I selected the colours green and red to represent charge and empty as the user will naturally understand the graph's key. One key issue I faced with the donut graph is that it didn't show an accurate numerical value like the design in Image1.2, I fixed this by using library chart.js. This library has a function that allows the user to select one part of the graph and it will display the numerical value to the user. The outcome is shown in Image 1.3, the graph is letting the user know that the Dining Room's battery is at 98% charge.

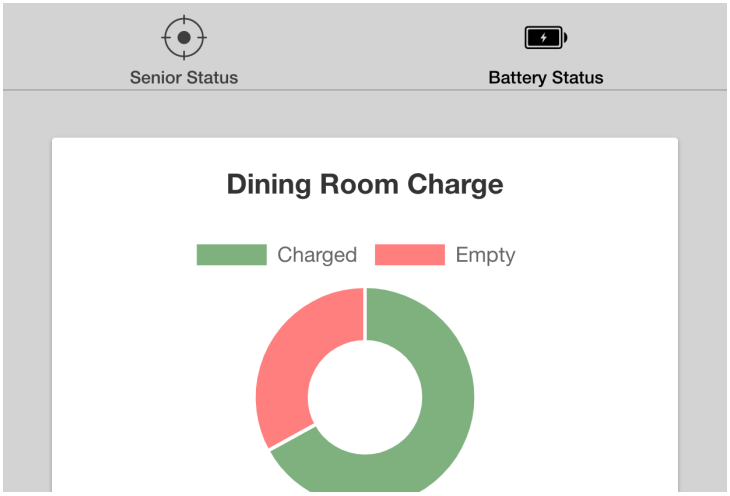
Tab layout

Another UX decision I made was to separate the two pages, (Senior Status and Battery Status,) into a two tab formatted application. This way the user had no way of getting confused where they were in the app and could always have a quick and easy way to decide which page they would like to be on. The layout is shown in the images below with the Tab selector on the bottom, closest to the user's thumb.





An alternative I was considering was having a page layout with buttons to show where the user was but this way the user has the pictures to show them which page they are on and will not have to find the buttons to navigate between pages they can simply select the tab they want.



This was the other alternative I faced, selecting the location of the tab bar. I decided to choose the bottom as the user will usually interact with the app using their thumb. Having the tab on the bottom of the phone is the most convenient location for the user to select the tab as they can reach it very easily with their thumb and won't have to rearrange their grip on the phone to select the tab.

Issues with the System

Network connection not working (Phone to Broker)

As the system depends on the phone being connected to a stable network connection, there is a high chance that a network error will occur during run-time. During this time the system may seem to be working, but will not receive any messages, so to the user, it will seem like the senior is not moving at all.

This behaviour is bad as it will mislead and stress the user. A good solution would be to alert the user as soon as the connection to the broker has been lost and let them know that no messages are incoming from the sensors. This will allow the user to fix the network error and not stress about the lack of movement from the senior.

Network connection not working (Sensor to Broker)

This error is much like the first one as it depends on the reliability of the network connection. A network failure with the sensors would likely cause a more serious issue as the user of the application would have no way of fixing the error on their end. At the moment the error would look like the senior is picking up no movement and the battery percentage of the sensor would most likely be a null value or zero.

The only outcome would be extra stress for the user in the worst case as they would think the senior is not moving at all. They would soon realise there is an issue with the sensors when they check on the senior so the issue is short term. A fix could be for the system to pick up on a network failure on the sensor's end and alert the user that there is a failure at the senior's residence.

Having a pet

Another major issue that could mislead the user is if the senior citizen had a large pet, say a dog or a cat. As the pet moved around the house the system would alert the user of movement and give them incorrect information about movement throughout the house. The senior may be in trouble in one room and not able to move to contact help, and their pet is moving around the house setting off sensors letting the user think the senior is fine.

The system has no current way of dealing with this, as there is no way of knowing how accurate the sensors are there would need to be a large planning process behind this source of error. The best way the system could deal with this is by being able to recognise which movement is from a pet and which is from the senior, there are a number of solutions to this that are either as complex as image recognition or something as simple as where the sensors are placed.

Senior leaving the house

The senior leaving the house is also a major source of error for the system. As the system will just keep sending no movement reports from the sensors, the user on the application side will just assume the senior is not mobile. This is misleading and could be fixed very easily with a sensor outside the residence that alerts the system when the senior has left and come back to the property.

UX Design

Key Difference

The key difference I found between the two kinds of applications is how the application is updated. IoT apps will most likely be updated in real time, changing the display every time an external provider sends some data to them. This differs from non-IoT apps as they will be updated generally from the user's interactions with input fields and buttons throughout the app. I have discussed this key difference and how they change the UX design of each app in the sections below.

Design of IoT Applications

Internet of Things (IoT) applications are created to interact with other devices through network connections meaning there are very limited human-to-device interactions. This will directly affect the user experience design of the application as the app will be designed to handle real-time inputs from the external provider and display an output on a device as soon as the data is received instead of waiting for user inputs.

As the main focus is on displaying data in real time, the design of the system will have very little user inputs and will most likely have very efficient ways of displaying the information to the user. I chose to display the data in graph format in my application but it can be displayed in many different formats such as a table, chart, text, image, etc.

With very little user input there will most likely be no buttons so that the user is shown as much data as possible without having to move through pages and perform actions to find or update the data. The data will usually be updated instead by event subscriptions that listen out for incoming data or changes in local variables.

Design of Non IoT Applications

The design of non-IoT applications I have found will usually be much more focussed on user input and interactions. As the display will generally not have to update until the user has input some data or actions, the system will not have to handle event subscriptions and can listen out for user actions. These will most likely be button or mouse selections.

These apps will still have a way of displaying data to the user but will have a lot more conditional statements and decision making in the back end of the app. These will decide what data is displayed to the user and when it will be displayed, depending on their inputs.