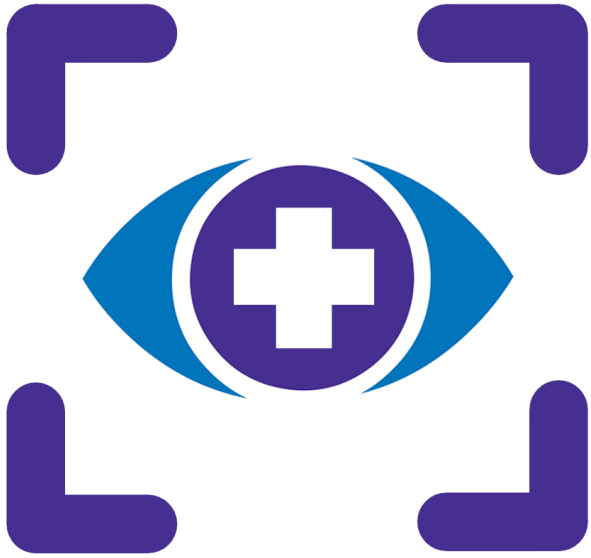


Health Data Insight 2021 Intern Project

A Prototype For Patient, Implant, and Anaesthetic Record Data Collection in Operating Theatres

Katy Higton
Jaskirat Kaur
Alex King
Laith Marsden
Tomas Pickford
Noo Rashbass
Jia Xiu Sai
Louise Treacy

Contents

1 Introduction

TheatreCapture uses scanning technology to document procedures in operating theatres. It automates data recording from anaesthetic machines, records patient and implant information, and links these in the cloud. This will enable identification of patients who may have received a faulty or expired implant, which will in turn increase patient safety. Key information will be recorded using Bluetooth beacons to note the location of the operation and an app that scans codes to link the information about a medical implant to who has received it. We have also found a way to alleviate the workload of anaesthetists by collecting data from the anaesthetic machine and storing this with the patient details. With the hands-off method of recording information, nurses and doctors will have more time to engage with their patients. By using existing technology, we offer a cheap and long-term solution that will help bridge the digital gap evident in healthcare.

1.1 Project Overview

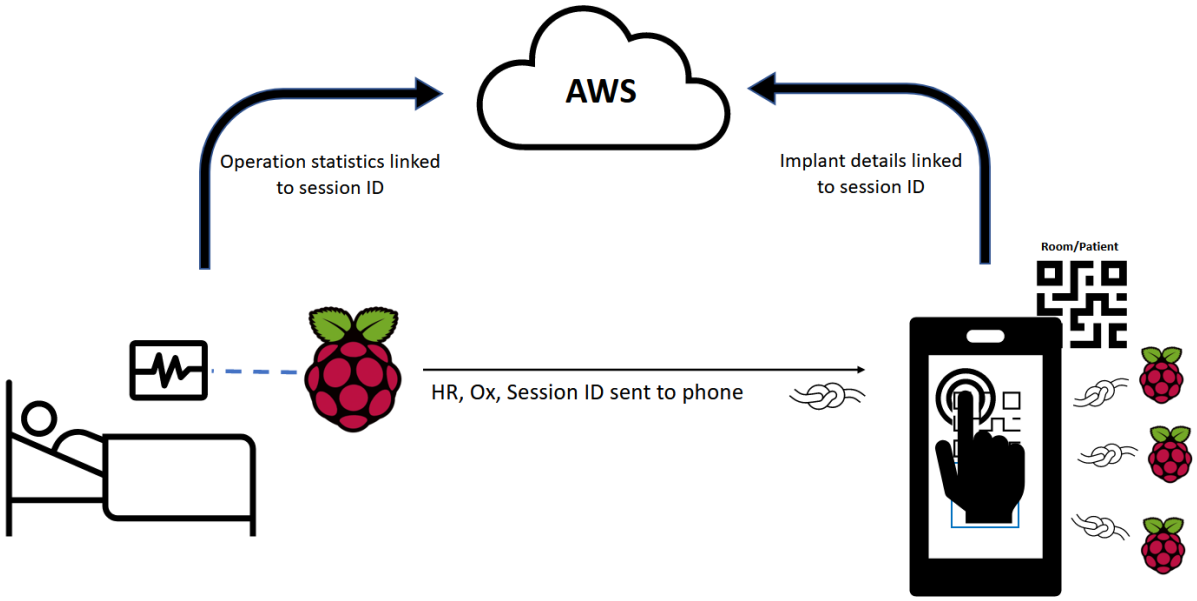


Figure 1: Technology Breakdown

The TheatreCapture project can be broken down into four main parts:

- The anaesthetic machine
- The app
- The cloud
- The marketing

The following sections will go in depth about each of these parts and possible future progressions.

1.2 Procedure Summary

1. Data is extracted from the anaesthetic machine using a Raspberry Pi.
2. The Pi sends this data in real time to be stored in the cloud along with a session ID.
3. It also sends this data via Bluetooth to any devices that connect to it.
4. The app initially connects to all the Pis it can hear.
5. The user selects their machine, and the app disconnects from the rest of the Pis. The phone records the session ID received from the selected machine.

6. The app is then used to scan:
    - The room code
    - The patient wristband
    - Any implants being used
7. The app sends this information to the cloud as it is scanned along with the session ID it received
8. The session ID can then be used to link up phone and Pi data in the cloud.

2 Data From The Anaesthetic Machine

Currently, a Raspberry Pi Zero W is used to extract data from the machine, then share it via Bluetooth and to the cloud. The Pi zero is the cheapest Raspberry Pi model with WiFi and Bluetooth capabilities, and has sufficient computing power for our needs.

2.1 Extracting Data From The Machine

2.2 Sending Data Via Bluetooth

Data is shared from the Raspberry Pi to the phone using the Bluetooth GATT protocol. The Pi is the GATT server, and the phone is the client - the phone connects to the Pi and reads data from it. The server contains one service, which has three characteristics - session ID, heart rate (HR), and oxygen saturation (SpO2). Heart rate and SpO2 are notifying characteristics, which means they update any connected devices with new values. Currently, the server does this every five seconds. The session ID is read only, so connected devices have to read again if they want to check their session ID is up to date. Ideally, the server would notify on session ID change. The code for the GATT server is written in python, and is available on GitHub.

For details of how the phone reads data from the Pi GATT server, see section 3.2.

2.3 Sending Data To The Cloud

2.4 Anaesthetic Record - Future Progression

3 Mobile Application

The mobile application is written using JavaScript and React Native, as some team members already had JavaScript experience. Additionally, React Native is open-sourced, allows for cross-platform development which reduces development time, and is widely used ensuring support for the library into the future. The code for the app can be found in intern2021-react-native.

3.1 Code scanning

The app acts as a scanner that records information about the patient, operating theatre, and implants used in the cloud. It uses the `expo-camera` module as it has built-in code detection.

3.1.1 GS1 Data Matrix

Global Standard 1 (GS1) is a not-for-profit organisation that develops and maintains global standards for business communication. GS1 standards in codes like bar codes, QR codes and data matrix codes are the most widely used system of standards in the world. GS1 Application Identifiers (AIs) are 2, 3 or 4 digit numbers which define the meaning and the format of the data that follows. This is important because implants will come with GS1 data matrix codes that encode their serial number, batch number, expiry date, etc. which can be used to flag implants that are expired or from a bad batch.

3.1.2 NHS Patient Wristbands

NHS Patient Wristbands also use GS1 Data Matrix codes, but they only encode 13 possible fields, with 4 of them specifically for newborn infants. These codes have a built-in checksum to ensure that the NHS

number is valid.

NHS number checksum
The method to check if an NHS number is valid is as follows. An NHS number is 10 numbers in 3-3-4 format (xxx-xxx-xxxx), for example: 943 476 5919.

1. Multiply each of the first nine digits by 11 minus its position.
   - 9 is the first digit: $9 \times (11 - 1) = 90$
   - 4 is the second digit: $4 \times (11 - 2) = 36$
   - 3 is the third digit: $3 \times (11 - 3) = 21$

     $\vdots$
   - 1 is the ninth digit: $1 \times (11 - 9) = 2$
2. Sum them all together,
$$90 + 36 + 24 + 28 + 42 + 30 + 20 + 27 + 2 = 299.$$
3. Get the remainder of the sum divided by 11,
$$299 \pmod{11} = 2.$$
4. Subtract the remainder from 11 to get the checksum,
$$11 - 2 = 9.$$
5. If the last number is not 10 and matches the last digit of the NHS number, it is valid.

3.1.3 Room Codes

The room codes are QR codes containing the ANANA code of the organisation followed by the operating theatre number.

3.2 Bluetooth Communication With The Anaesthetic Machine

Bluetooth connection between the Raspberry Pi GATT server and the phone are handled in the `BleList.js` component of the app, making use of the `react-native-ble-plx` package. Details of the Bluetooth communication from the Pi side are in section 2.2.

The app connects to all the Pi GATT servers it can hear, and displays them in a list ordered by proximity. Also in the list are heart rate and oxygen saturation readings from the machine. Other vital signs could also be displayed if desired. The user can then make a comparison between the machine in front of them and the readings on the app to select the correct machine.

Proximity to a machine is calculated using the Bluetooth signal intensity (RSSI) recieved from it. Signal intensity values fluctuate quite significantly [1], so multiple measurements are made and an average is taken over a period of 5 seconds. A longer averaging period would lead to more reliable results, but would make the app slower to update if the user moves around. Therefore, a balance needs to be found between reliability and speed. More testing needs to be done on this.

Before the app can start scanning for Bluetooth devices, a new instance of the BleManager class from `react-native-ble-plx` must be created. Then it begins scanning, and connects to devices with the UUID of our GATT server, `820fdfb2-1648-4963-aed4-dee794022118`. Once connected to a device, the app reads its session ID, and begins monitoring the HR and SpO2 characteristics. In addition, a loop starts that measures the RSSI value from the device every 0.5 seconds, and averages these readings every 5 seconds. These loops run asynchronously for each device that connects, and the order of the list updates every time a new average is calculated for a device.

Once a machine is chosen, the app disconnects from all the other devices. The RSSI averaging calculations are stopped for all devices. If alternatively the user chooses to proceed without a machine, all devices are disconnected. Whether a machine is chosen or not, the user has the option to search again for Bluetooth devices. When a rescan is initiated, the old BleManager is "destroyed", and a new instance of the class created. This may not be necessary, and the code could be changed to use the same instance the whole time the app is running.

3.2.1 Known Bugs

There is currently a bug where the app stops measuring RSSI values from the devices it is connected after scanning for a period of time on the order of 30 seconds to a minute. The `react-native-ble-plx` package makes heavy use of promises, and the function to measure the RSSI of a connected device, `readRSSI()`, returns a promise. After some time and for an unknown reason, the app stops resolving these promises, causing a large number of pending promises to build up, which eventually crashes the app. This has been partially resolved by cancelling promises if they take too long to resolve. However, this still leaves the problem that RSSI values stop being measured after around a minute of scanning, which means the ordered list is no longer being updated. This is most likely a bug with the BLE package we are using, so the solution may be to deal with Bluetooth connection for Android and iOS natively.

3.3 User Interface

The app uses React Native Elements UI toolkit. This section will go through each component of the app and explain its functionalities.

Machine Menu
The machine menu handles the connection and selection of Pis. This component is stored in `MachineMenu.js`.

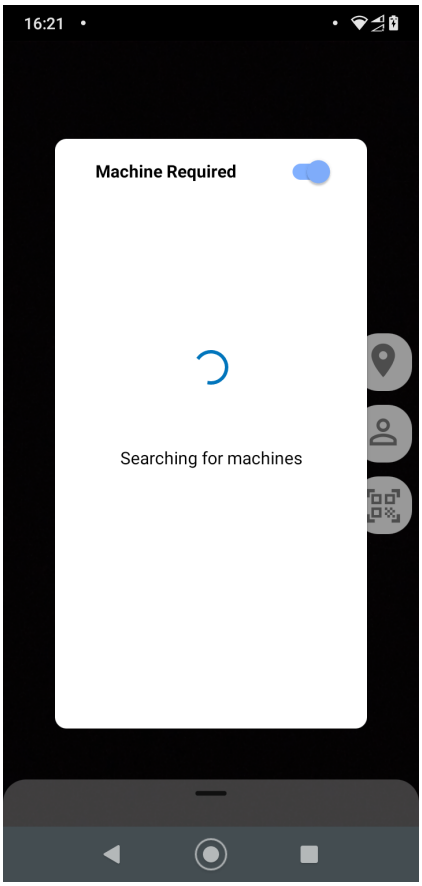


Figure 2: Machine Menu Searching

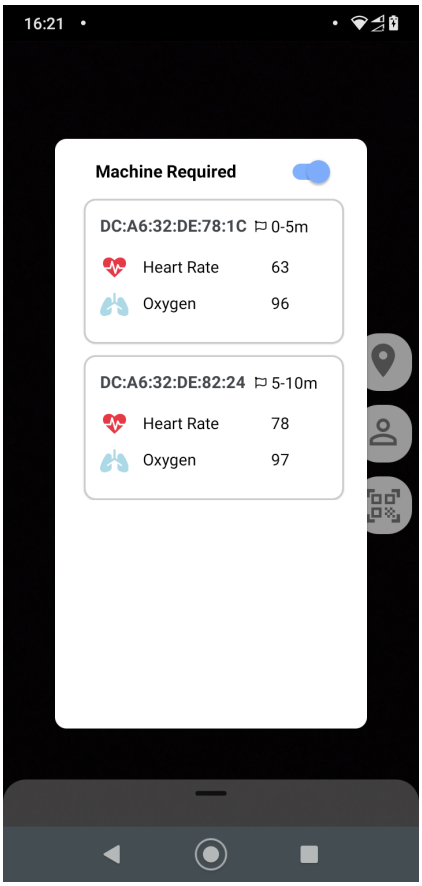


Figure 3: Machine Menu Options

Upon opening the app, the machine menu will initialise automatically, searching for any nearby Raspberry Pis broadcasting data from the anaesthetic machines as shown in Figure 2. Once one or more Pis are detected, the cards will appear displaying the characteristics of the Pis as shown in Figure 3.

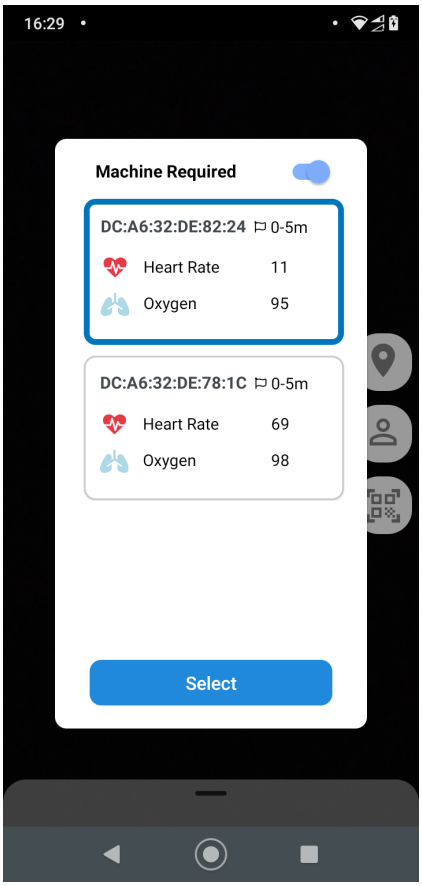


Figure 4: Machine Menu Selected

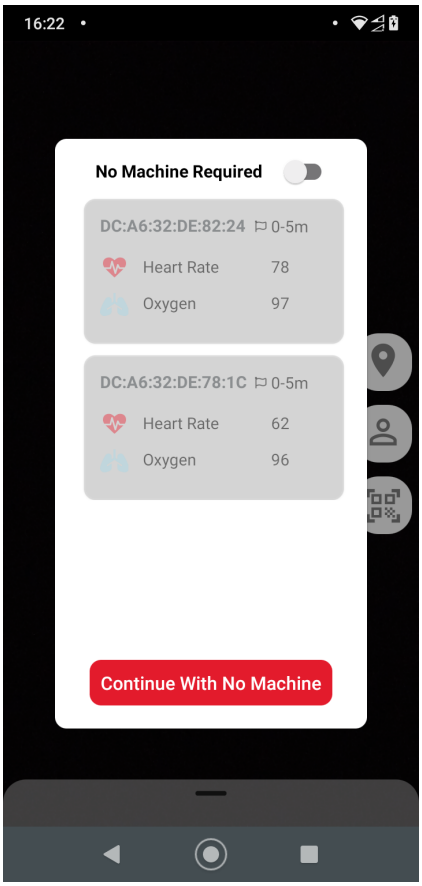


Figure 5: Machine Menu Disabled

The user can then press on the machine that needs to be connected to and the option of "Select" will appear as shown in Figure 4. If no machine is required, the toggle on the top of the machine menu can be pressed for the option of "Continue With No Machine", which will appear as shown in Figure 5.

Top Sheet
The top sheet displays the current characteristics of the Pi selected.

Figure 6: Top Sheet with Pi Connected

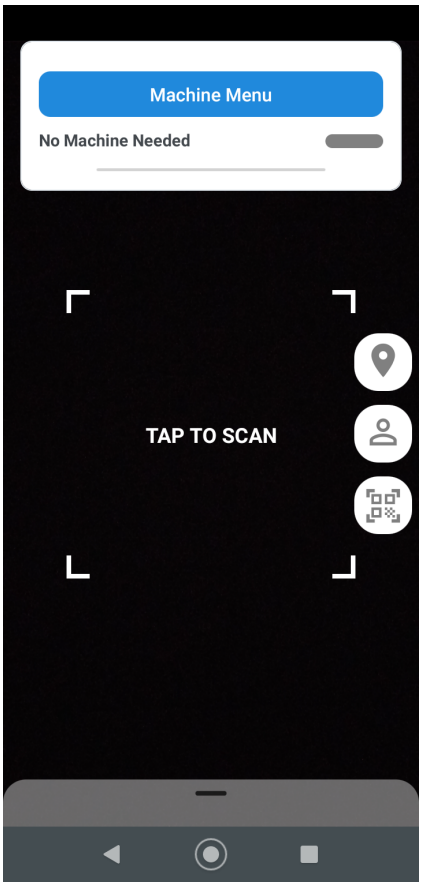


Figure 7: No Pi Selected

Once the user presses the "Select" button in the machine menu, the machine menu will be replaced by a top sheet, that is partially hidden. The user can reveal the top sheet by tapping on the handle bar and the top sheet will be fully revealed, displaying the live readings from the anesthetic machine as shown in Figure 6. However, if the option "Continue with no machine is selected", the machine menu will be replaced by a top sheet without any characteristics as shown in Figure 7.

Bottom Sheet

The bottom sheet displays the codes scanned. This component is built using the bottom sheet package (Version 4.15) and stored in `CustomBottomSheetScrollView.js`.

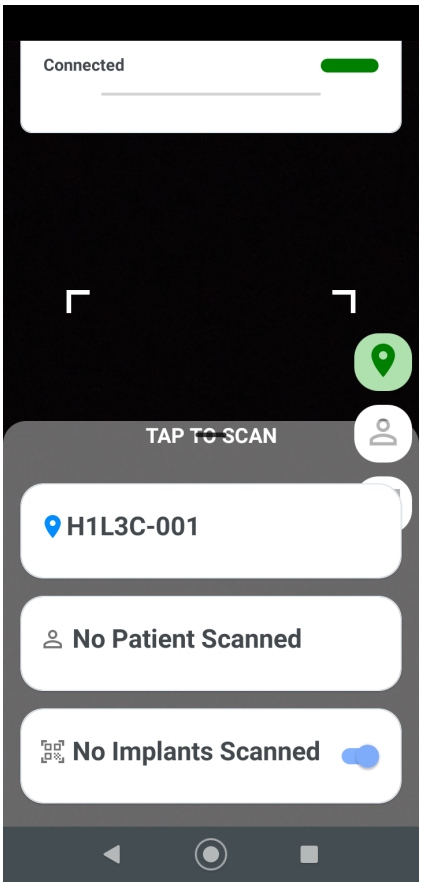


Figure 8: Bottom sheet with room scanned

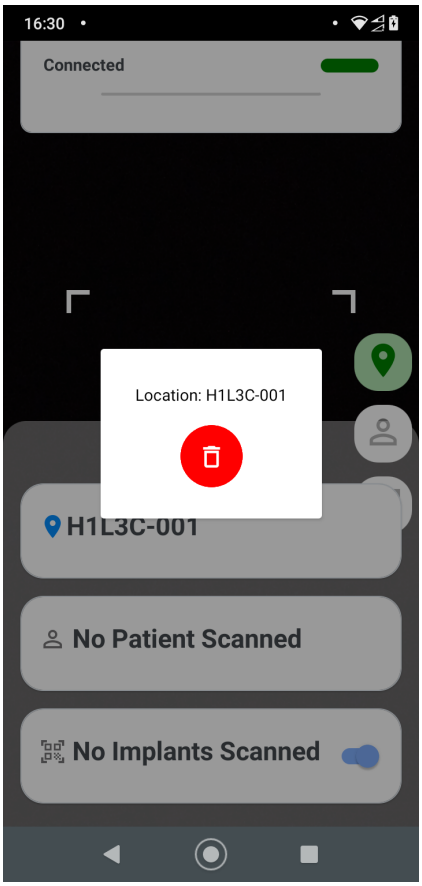


Figure 9: Overlay of scanned code displaying it's details

Once a code is scanned successfully, the data will be stored on the phone and automatically uploaded to the cloud. The icons on the right represent each different type of code and stored in `TrafficLights.js`. If a room code is successfully scanned, the room code will be recorded and the location icon will turn green as shown in Figure 8. Tapping on the item in the bottom sheet or the icons in the traffic lights will bring up an overlay with the details of the code scanned, with an option to delete from the record as shown in Figure 9.

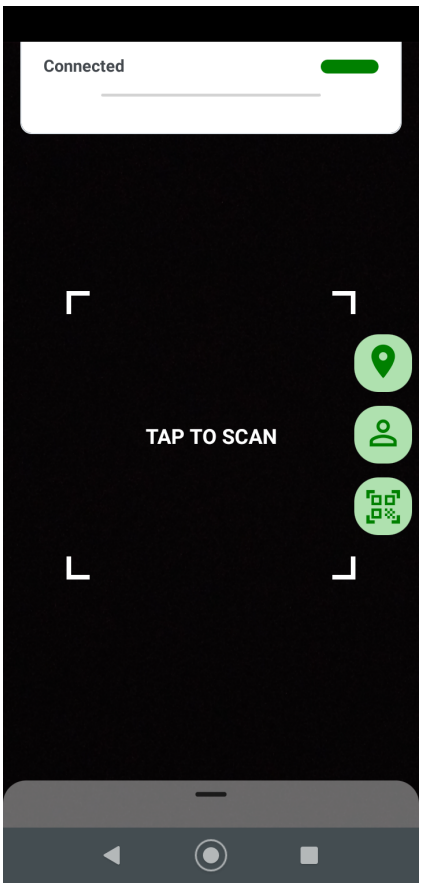


Figure 10: Successful icons

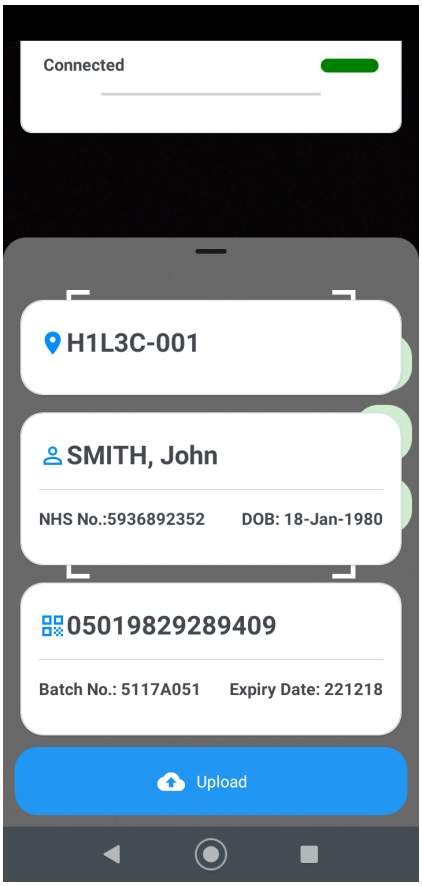


Figure 11: Bottom sheet with a code scanned for each type

If all three types of code are scanned, the icons on the right hand side of the app will turn green as shown in Figure 10. The bottom sheet can also be expanded to show a brief summary of all the codes scanned in the session. An "Upload" button will appear if all the required codes have been scanned, as shown in Figure 11.

3.4 App - Future Progression

4 Cloud

4.1 AWS IoT Greengrass

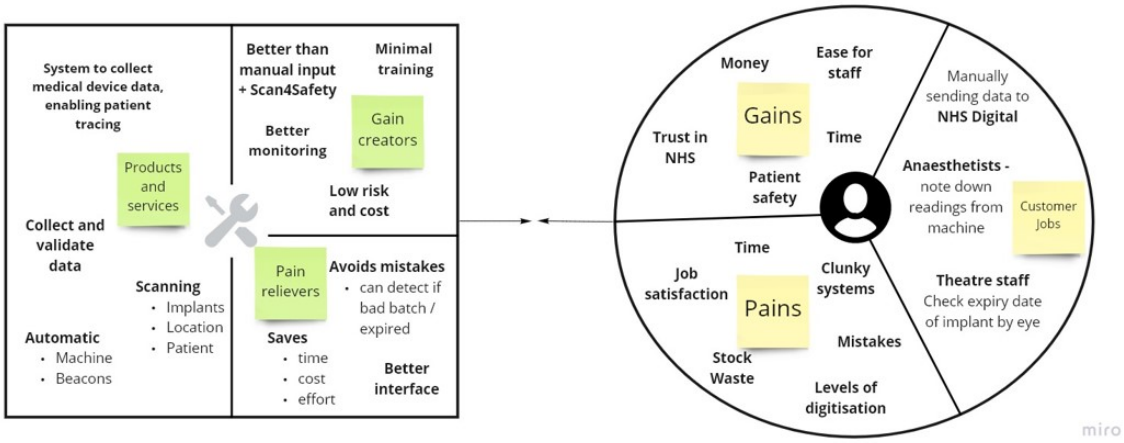
4.2 Pis to Cloud

4.3 Phone to Cloud

4.4 Cloud - Future Progression

5 Marketing

5.1 Value Proposition



5.2 Costing

5.3 USPs

6 GitHub Repositories

These repositories are owned by Health Data Insight GitHub account and some of them are private.

- React Native app: intern2021-react-native
- Extracting data from machine: VSCapture
- Broadcasting data from the anaesthetic machine: interns2021-greengrass-component
- Experimentation with geolocation with Raspberry Pis: intern2021-code-snips

References

[1] Eladio Martin, Oriol Vinyals, Gerald Friedland, and Ruzena Bajcsy. Precise indoor localization using smart phones. *MM'10 - Proceedings of the ACM Multimedia 2010 International Conference*, pages 787–790, 10 2010.