RollSmart - Capstone Project Final Report

SYSC4907, Group 33

By:
Corbin Garlough - 101101493
Isabelle Bryenton - 101077788
Kenny Deng - 101122713
Mark Johnson - 101110080
Yunas Magsi - 101115159

Supervised by: Dr. Adrian Chan

Department of Systems and Computer Engineering
Carleton University
Ottawa, Canada
April 12, 2023

Contents

1 Introduction **5**
- 1.1 Background . . . 5
- 1.2 Short Term Goals . . . 6
- 1.3 Long Term Goals . . . 6

2 System Overview **7**
- 2.1 Microprocessor . . . 8
- 2.2 Powering The Rollator . . . 11
  - 2.2.1 Witty Pi Power Management . . . 13
- 2.3 Component Mounting Hardware . . . 15
- 2.4 Cloud Data Storage Back-end . . . 15
- 2.5 Data Analysis Application . . . 17
  - 2.5.1 Proposed Application Layout . . . 19
  - 2.5.2 Real Time Application Layout . . . 20
  - 2.5.3 Future Implementations For Application . . . 26

3 System Design **27**
- 3.1 Speed and Distance . . . 27
- 3.2 Heart Rate Monitoring . . . 28
- 3.3 Pace Measuring . . . 29
- 3.4 Seat Occupancy Detection . . . 29
- 3.5 Handlebar Strain Sensor . . . 30

4 Implementation **32**
- 4.1 Implementation Plan . . . 32
- 4.2 Strategy . . . 32
- 4.3 Completed Implementation . . . 32
- 4.4 Data Acquisition . . . 34
  - 4.4.1 Pushing sensor data to database . . . 34
  - 4.4.2 Sensor Progress Update . . . 35
  - 4.4.3 Final Implementation and Installation of Sensors on RollSmart . . . 38
- 4.5 Testing . . . 45
  - 4.5.1 Unit Testing . . . 45
  - 4.5.2 Sample data for data analysis testing . . . 46

5 System integration **46**
- 5.1 3D Printed Sensor Mounting Hardware . . . 46
- 5.2 Continuous Integration, Delivery, and Deployment (CI/CD) . . . 47
- 5.3 RollSmart Test Fixture and integration into GitHub . . . 48

6 Project Management **48**
- 6.1 Requirements . . . 48
  - 6.1.1 Abstract Requirements . . . 49
  - 6.1.2 System Requirements . . . 50
- 6.2 Justification of suitability . . . 51
- 6.3 Team Member Tasking . . . 52
- 6.4 Milestones . . . 52

6.4.1 Configure Raspberry Pi and connect sensors . . . 52
6.4.2 Read data from sensors . . . 53
6.4.3 Sensor mounting . . . 53
6.4.4 Sensor data export . . . 53
6.4.5 Data Summary generation . . . 54
6.4.6 User test . . . 55
6.5 Project Timeline . . . 56
6.6 Budget . . . 58

7 Risk Analysis **59**
7.1 Project Delivery . . . 59
7.2 Team Member Personal Safety . . . 61
7.3 Functional Safety . . . 61

8 Conclusion **63**
8.1 Future Directions . . . 63

List of Figures

1 Nexus rollator . . . 6
2 System overview . . . 7
3 RollSmart System Diagram . . . 8
4 Raspberry Pi 3 Board . . . 9
5 Raspberry Pi Zero W Board . . . 10
6 Tentative I/O Schematic for Raspberry Pi Zero . . . 11
7 Tentative I/O Schematic for Raspberry Pi 3/4 . . . 11
8 Lithium Polymer battery Pack . . . 13
9 Witty Pi Board . . . 14
10 Ender 3 Pro 3D Printer and the Cura environment . . . 15
11 Proposed Firebase Organization Structure . . . 16
12 RollSmart Desktop Application Structure . . . 18
13 User Authentication Page . . . 19
14 Sample Patient List . . . 19
15 User Information . . . 19
16 Graphical Representation . . . 19
17 Authentication Page . . . 20
18 Practitioners List for User Selection . . . 21
19 User Dashboard . . . 22
20 User Detailed Analytic Selection . . . 23
21 User Graph Output . . . 24
22 New Account Creation . . . 25
23 Sample Account Creation Email . . . 25
24 Process flow of Rollsmart Application . . . 26
25 Littelfuse 59025-020: Magnetic Reed Switch . . . 28
26 MAXREFDES117: Heart Rate Monitoring Module . . . 29
27 Bosch 9-Axis BNO055 IMU Sensor . . . 30
28 NEXTION WI1802AX4+WI0728: Load Cell . . . 30
29 Daoki BF350-3AA strain gauge . . . 31
30 Diagram of RollSmart handlebar Sensor placement . . . 31
31 Status of project tasks over time . . . 33
32 Status of project tasks as of Feb 27 2023 . . . 33
33 File tree of RollSmart scripts . . . 34
34 Process flow for Initializing Rollsmart activity and collecting data from sensors . . . 35
35 Python code snippet of push-data function . . . 36
36 Process flow for sensor data upload to Database . . . 36
37 Procured sensors . . . 37
38 FlexiForce Sensor implementation . . . 37
39 Hardware Sensor Sampling Console Logging . . . 38
40 Final Rollsmart Product . . . 39
41 Rollsmart Hardware Configuration . . . 39
42 Final Wiring diagram for completed implementation of RollSmart . . . 40
43 Final Physical Wiring Configuration . . . 40
44 Reed Switch Sensor Location . . . 41
45 Reed Switch Magnet Interaction . . . 41
46 Heart Rate Monitor Handlebar Location . . . 42

47 Top Mounted Heart Rate Monitor on Handlebar . . . 43
48 Bottom Mounted Heart Rate Monitor on Handlebar . . . 43
49 IMU Central Mount Location . . . 44
50 Strain Gauge Mounted on Handlebar . . . 44
51 Load Cell Mounted Beneath Seat . . . 45
52 Heart Rate Sensor Handle and Reed Switch Mounts . . . 46
53 Heart Rate Sensor Handle and Reed Switch Mounting locations . . . 47
54 Battery Pack and Raspberry Pi Housing . . . 47
55 Microprocessor configuration milestones . . . 53
56 Sensor Mounting Milestones . . . 53
57 Database Creation Integration to data acquisition Solution . . . 54
58 User Data Summary Generation milestones . . . 54
59 User Test Milestones . . . 55
60 RollSmart Project Timeline . . . 56

List of Tables

1 Component Requirements . . . 8
2 Boards under consideration . . . 9
3 Power requirements for all powered hardware on Rollator . . . 12
4 Abstract Requirements Specification . . . 50
5 System Requirements Specification . . . 51
6 Team Member Tasking for Project Deliverables . . . 52
7 Milestones, Leads, and approximate completion date . . . 57
8 Bill of Materials for RollSmart . . . 58
9 Classes of Risk Probability . . . 59
10 Classes of Risk Severity . . . 59
11 Classes of Risk Impact . . . 59
12 Risk Identification . . . 61

SYSC 4907 Final Report: RollSmart

C. Garlough, I. Bryenton, K. Deng, M. Johnson, Y. Magsi

1 Introduction

1.1 Background

Canada's senior population (aged 65 and older) is projected to double by 2036 [3]. For this aging population, the risk of falling every year increases from approximately 32 percent at age 65 to over 40 percent for 70 and over [8]. Visual and balance impairments are key factors that contribute to elderly fall incidents [6]. Furthermore, medication for illnesses common in this age population (diabetes, anti-epileptics, etc) makes the person more prone to falling due to the side effects [6]. Therapeutic devices such as Rollators, shown in Figure 1, have been shown to decrease the risk of falls in elderly users by improving their gait mechanics [7]. Another use for rollators is physical rehabilitation. Many injuries to the lower body can make movement difficult for the person due to pain when weight is applied to the injury, so having a device to assist in load bearing removes some inconvenience.

Premature mortality and health issues among the elderly have been linked to inactivity [2]. As the senior population continues to grow, the healthcare system's demand will be more stressed and challenged. To reduce the demand on the healthcare system by unhealthy and inactive seniors, they should be made aware of healthy habits and activities. The use of a rollator is linked to promoting greater health and welfare in addition to granting the ability for improved mobility and independence [2].

Current mobility aids offer physical support to the user but lack the ability to provide feedback on if the device is being used properly to benefit the user. The user can be either a person receiving physical rehabilitation or an elderly person that requires assistance in day-to-day activities on a more permanent basis. During medical appointments, it is the responsibility of the user to report how they have been using the device, but people make generalizations or exaggerations that can negatively affect how the medical professional advises the user to alter or continue using the mobility aid.

Prior art in the domain of computer-integrated mobility devices is extensive, with a broad variety of different objectives having been explored. Among the most relevant to this project, the University Institute of Lisbon [9] developed a physical rehabilitation assessment system based on a rollator sensor suite measuring force, acceleration, and motion, to detect balance and stability concerns for users. Kulyukin et al. [4] prototyped a smart rollator platform with an onboard navigation system based on the movement of the user, designed to assist elderly members of the population with performing wayfinding tasks using a visual and auditory computer interface. Lastly, the Institut des Systèmes Intelligents et de Robotique [8] conducted a review of a variety of smart rollator prototypes to develop a general methodology for developing new smart rollators, with particular attention paid

to sensor choice and system integration to create an optimized medical device.

Figure 1: Nexus rollator

1.2 Short Term Goals

The short-term goal of this project is to attach sensors and a microprocessor to a rollator in order to collect data about how the usage of the rollator by the user. A medical professional will be able to monitor the user's condition or adjust rehabilitation on a more continuous basis using collected data streamed to a cloud service for remote access. The RollSmart will focus on data collection similar to the work done at the Institute of Lisbon, and while having any navigation assistance is outside the scope of this project, the paper from Kulyukin illustrated how inexpensive sensors can be integrated into a useful piece of technology to improve the lives of its users and is able to be recreated without any very specialized equipment.

1.3 Long Term Goals

Thinking about where the project is heading in the future, additional testing and refining should be done to fine-tune the accuracy of the system, and more features for data collection can be included to form a more complete picture of the world around the rollator. These features could include safety items like ledge detection or collision warnings, or navigation assistance devices. The data-viewing application can also be expanded to incorporate data from other systems to create an all-in-one health monitoring application.

2 System Overview

The Rollsmart system has been designed in accordance with the requirements specified in subsection 6.1. The requirements outlined there will be met by implementing the following hardware and software components as shown in the system diagram in Figure 2. These components will interface with each other as illustrated in Figure 3.

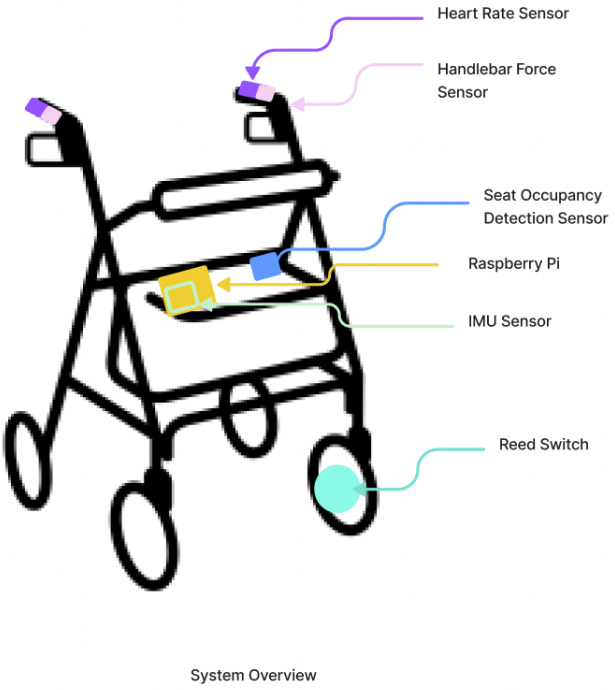

Figure 2: System overview

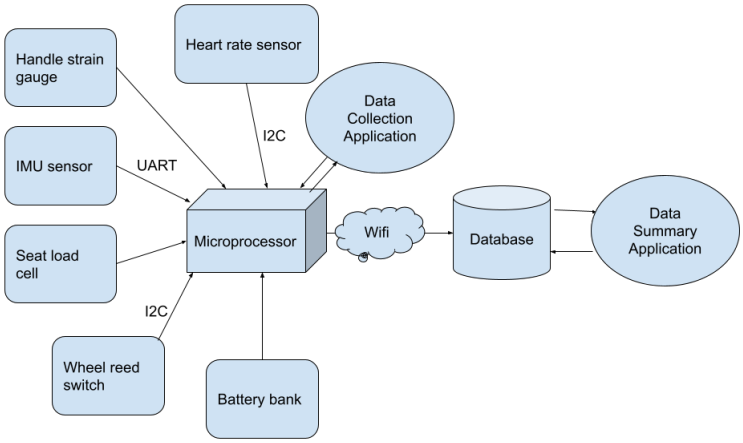

Figure 3: RollSmart System Diagram

2.1 Microprocessor

Requirement: The RollSmart must be equipped with a controller board that can interface and collect data from all of the sensors, real-time clock, and WiFi connectivity. The microprocessor chosen for the RollSmart is a Raspberry Pi. After extensive research on various micro-controllers and microprocessors. It became clear that the Raspberry Pi offers a substantial amount of GPIO pins, computational power, WiFi/Bluetooth connectivity, can be used for multi-threading operations more simply, and many more features. The Pi offers the best performance and flexibility. The Pi has a built-in real-time clock, which a lot of other microprocessors do not offer built-in. The Raspberry Pi runs its own operating system which is Linux based, meaning we can use the device as a regular computer if desired.

Before assessing the microprocessor options, it was important to understand the GPIO requirements of the sensors outlined above. Table 1 outlines the chosen sensors and their GPIO requirements.

Sensor	**Data Output**	**Digital Inputs**	**Power**	**Implementation requirements**	**Quantity**
BNO055 (IMU) Adafruit Breakout	SCL, SDA (digital I2C)	INT, ADR, RST, PS1, PS0	5V in , GND	Since we are using Adafruit breakout, it can be wired directly to the pi.	1
Reed Switch	SCL, SDA (digital I2C)	11		I2C	2
MAXREFDES117 HR Sensor	Analog Vout	n/a	3V, GND	Analog sensor which needs to be connected to ADC	2
NEXTION WI1802AX4+WI0728 Strain Gauge	D_0	RST	3V, GND	Ultra long-distance wireless antenna	1

Table 1: Component Requirements

To begin, various microprocessors were evaluated to determine which one would most efficiently

set out the requirements set above, specifically considering the GPIO requirements detailed in Table 1. A summary of the boards considered is shown in Table 2. The Adafruit huzzah32, NodeMCU, and ESP32 SX1278 (LoRa) boards were decided against since they did not meet the GPIO requirements and the group's experience using the other board options. The Raspberry Pi's proved to be the solution that would meet all the requirements while facilitating the development of software since it can run Python scripts. Two versions of the Pi's were considered, the Raspberry Pi 3 and the Raspberry Pi Zero W (Figure 4) and (Figure 5).

Device	**Wifi**	**I2C**	**Analog GPIO**	**Digital GPIO**	**Pros**	**Cons**
Adafruit Huzzah32	Yes	no	5	8	A lot of public libraries	Limited GPIO, lack of team experience with device
NodeMCU	Yes	yes	0	11	USB for Arduino Programming	Requires it's own power source Does not natively support python
Raspberry Pi 3	Yes	Yes	0	40	Microprocessor which runs Linux. Easily execute python scripts, easy to use given groups experience. Has multiple I2C lines	No analog pins, will require use of ADC.
Raspberry Pi Zero 2 W	Yes	yes	0	40	Multiple I2C & 3.3V pins, Same GPIO as full size Raspberry Pi 3/4 with a smaller footprint which requires less power. Python support	No analog pins, will require use of ADC
ESP32 SX1278 (LoRa)	yes	yes	0	6	Ultra long-distance wireless antenna	Not enough GPIO
Arduino MKR1000 Wi-fi	Yes	yes	8	8	Supports I2C, Small form factor, has analog input pins	Does not support python, Not enough digital I/O

Table 2: Boards under consideration

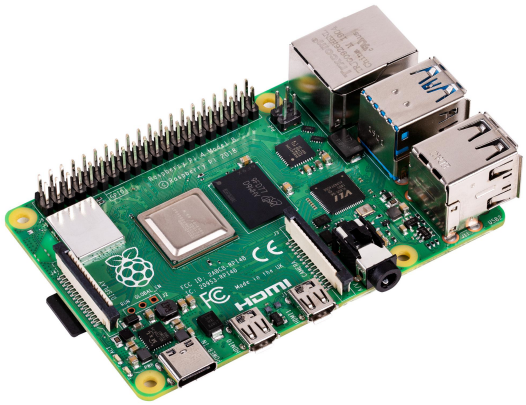

Figure 4: Raspberry Pi 3 Board

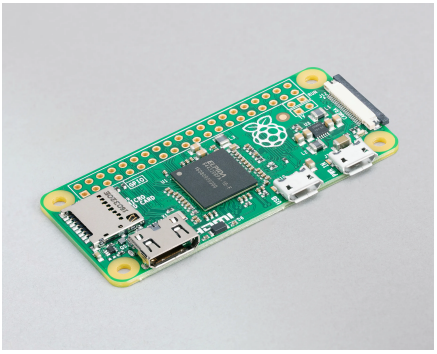

Figure 5: Raspberry Pi Zero W Board

There are a multitude of benefits to the Raspberry Pi 3 which wouldn't fully be exploited in the RollSmart. Its power requirements increase likely wouldn't be able to be sustained for the 2-and-a-half day power requirement. For this reason, the design was revised to be implemented using a Raspberry Pi Zero W. A smaller version of the Raspberry Pi 3, which has the same amount of digital GPIO pins while requiring much less power. The Pi Zero W has fewer USB ports, less RAM, and a single-core CPU with less processing capabilities than the raspberry pi 3. The requirements of the microprocessor were reviewed and Group 33 has determined that the Pi Zero W will have the sufficient CPU power to accomplish our objectives, and greatly improve the battery life of the RollSmart.

I/O wiring diagrams for both Pi zero and Pi 3 were drafted and shown in Figure 6 and Figure 7. These diagrams demonstrate the circuitry which will be implemented in order to connect each sensor to the microprocessor, along with confirming that using a Raspberry Pi Zero will offer sufficient GPIO for the RollSmart requirements.

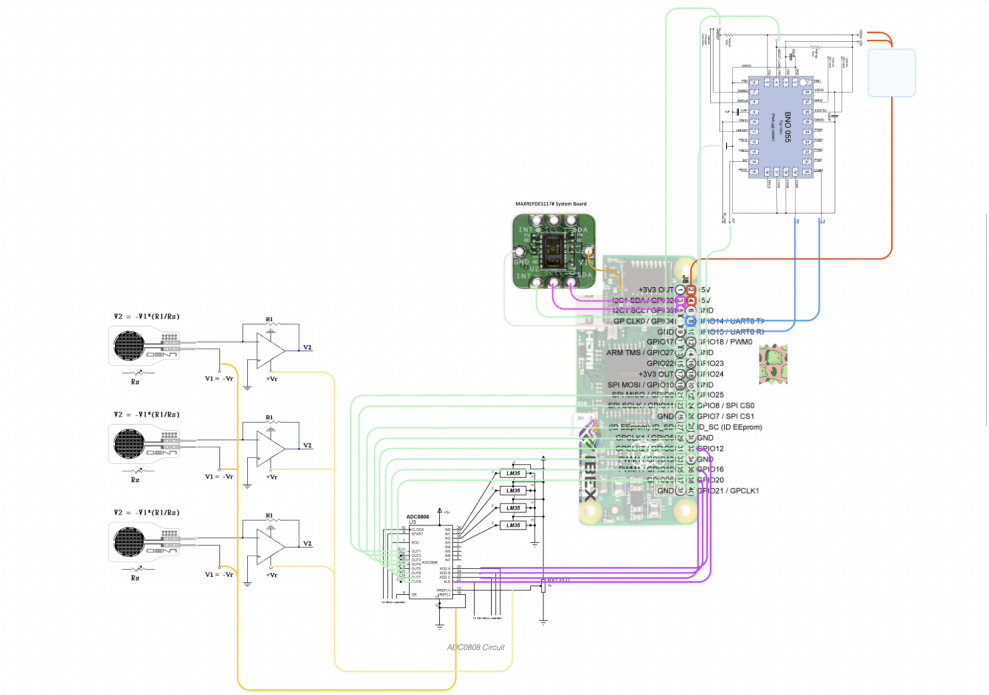

Figure 6: Tentative I/O Schematic for Raspberry Pi Zero

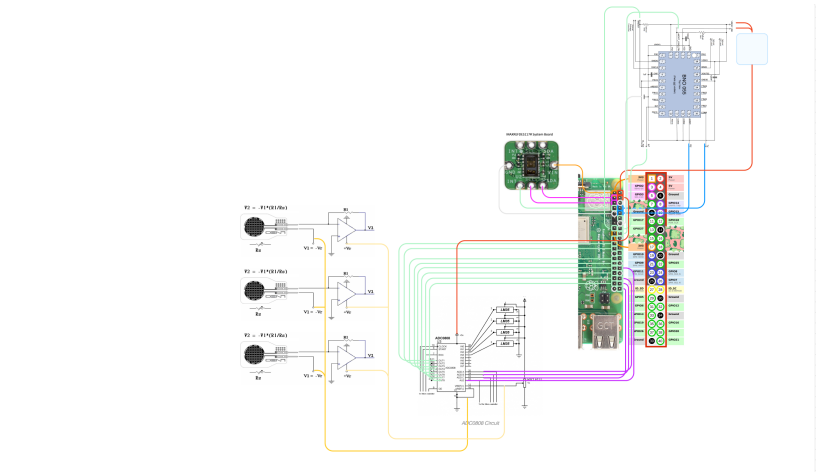

Figure 7: Tentative I/O Schematic for Raspberry Pi 3/4

2.2 Powering The Rollator

The rollator requires an independent power source to supply power to the microprocessor and the various connected sensors. An analysis of the power consumption of the desired sensors for the

rollator is shown in Table 3. What kind of battery and recharging system should it entail? We must consider form factors, ease of use, repairability, and upgradability. To conserve most of the battery that's stored

The most off-the-shelf solution is to use an existing Lithium Polymer portable battery bank product available on the market with a micro USB cable directly powering the Witty Pi and Raspberry Pi 4 microprocessor. One cable from the microprocessor to the power bank for supplying power. One cable from the power bank to the user for charging. This solution is very intuitive and can be easily modified in the future based on the increasing/decreasing power needs of the user. Portable battery banks are very readily available and easy to replace (batteries have a limited usable lifespan, and having a self-contained power unit makes them easy to replace). As a bonus, the portable battery bank comes with solar charging and additional USB ports. The calculation of the energy produced by the solar cells and Witty Pi is not considered in the calculations below. This is due to the matter that this consumption can vary vastly depending upon the usage per case. The additional USB ports, although not recommended, can be used in emergency situations by the rollator user in the event their cell phone is out of charge

An example of an existing readily available generic Lithium Polymer battery bank along with 36.6 Ah power capacity and input/output specs is shown in Figure 8. (36.6 Ah, Solar powered. 4 USB-A output (5V, 2.1A). 1 USB-C input/output (5V, 3A)).

Device	Quantity	Power Consumption			
		Voltage (V)	Current (A)	Power (W)	Quantity * Ah
Raspberry Pi Zero	1	5	3	15	3
Littelfuse 59025-020 Reed Switch	1	3.3	0.01	0.033	0.01
Maxrefdes 117 heart-rate sensor	2	2.4-3.6 (3.3)	0.0137	0.04521	0.0274
BNO055 IMU	1	3.3	0.01	0.033	0.01
NEXTION WI1802AX4+ WI0728 load cell	1	3.3	0.015	0.05	0.03
Daoki BF350-3AA strain gauges	2	3.3	0.01	0.033	0.02
Total					3.071

Table 3: Power requirements for all powered hardware on Rollator

Figure 8: Lithium Polymer battery Pack

For a worst-case scenario with maximum power draw and minimal idling from the Pi and sensor hardware, with an ∼36 Ah Lithium Polymer battery pack, we can expect ∼12 hours of use between charges. For an average mix of use of light use and power draw along with a moderate idle time we can expect ∼1.5 days of use between charges. For a best-case scenario with little use and power draw along with excessive idling we can expect ∼2.5 days of use between charges.

2.2.1 Witty Pi Power Management

The Witty Pi is a power management board designed for Raspberry Pi. This add-on benefits the overall power consumption for RollSmart. The Witty Pi has its own built-in microprocessor that can manage a real-time clock without the internet, power on/off the Pi, and connect to various sensors. The Witty Pi shown in Figure 9 offers certain advantages when integrated into RollSmart. These include an additional microprocessor, controlling the on/off state of the Pi, and having a real-time clock. The microprocessor in the Witty Pi can be paired with the IMU sensor on the RollSmart to determine when the user is actively using the RollSmart, by sensing movement. When the user has been inactive for a set time, determined by the IMU, the Pi can safely be powered off to sleep mode. As the RollSmart is an application that is used both indoors and outdoors, the real-time clock serves to provide the correct time for data logging, which will keep accurate logs of all sensors when the Pi is powered on and an internet connection is unavailable. Although the trade off includes additional cost and extra hardware being implemented in the system. This simple add-on feature by far improves the battery life, quality of the product, and reduces active sensor usage.

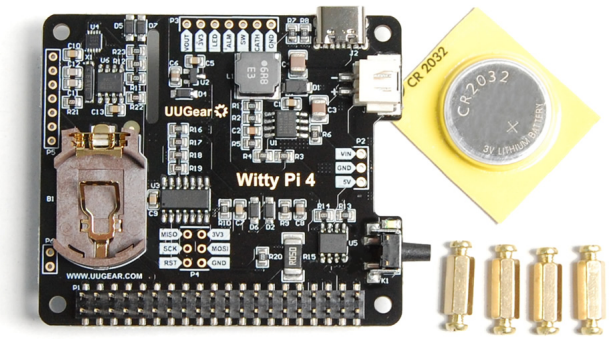

Figure 9: Witty Pi Board

For the electrical components, the Witty Pi is connected via USB type c to USB on the portable battery bank. It also comes with a secondary power source that keeps the internal clock operating. This is powered by a single CR2032 battery and the clock only requires a current of 4uA/h, which can last the same amount of time as a traditional watch. The Pi itself is powered by the Witty Pi. The Witty Pi is a shield that lies on top of the Pi and is connected by the GPIO headers. In turn, the Pi is powered through the 5V and GND pins from the Witty Pi. Therefore, the Pi does not need any additional power supply to power it. Making Witty Pi regulates all the power management. In terms of communicating between the Witty Pi and Pi, the communication takes place over the I2C SDA and I2C SCL GPIO lines. The built-in microprocessor of the Witty Pi can determine the active usage from the IMU sensor. These communication lines are used to send instructions to the Pi accordingly.

2.3 Component Mounting Hardware

Attaching all of the hardware to the rollator in a way that looks professional, securely holds components in place, and does not interfere with the use of the mobility aid is an important piece of the RollSmart project. The suite of sensors being equipped to the rollator includes sensors that are placed on or near critical areas of the rollator. The heart rate sensors are located on the grips, this is where the user applies their body weight and controls the rollator. The piece designed to attach the sensor needs to be comfortable to hold, and robust so that no loss of control results from an unstable grip. The reed switch is placed next to the rear wheel, and keeping the area where the user's feet are clear of any obstructions is vital given that users are already suffering from mobility impairments.

Component mounting is accomplished using 3D printing. 3D printing allows for the creation of complex parts that are difficult or impossible to achieve with traditional machining methods. This is very useful when creating mounting hardware that needs to conform to the shape of the rollator frame or accommodate a specific sensor or microcontroller. 3D printing also allows for rapid prototyping and creating many iterations of a part to find an optimal design. The printer being used is called an Ender 3 Pro and is an FDM 3D printer, it can be seen in Figure 10. This means it prints objects by extruding a melted plastic filament through a print head in one line. These lines of plastic are stacked to create 3D pieces. The tool used for creating 3D models of the rollator frame and mounting hardware is Inventor. This CAD (computer-aided design) program is a modeling environment that creates 3-dimensional objects from 2-dimensional drawings. These objects are then brought into another program called Cura, shown in Figure 10, which takes the 3D object and outputs a path for the head of the printer to follow.

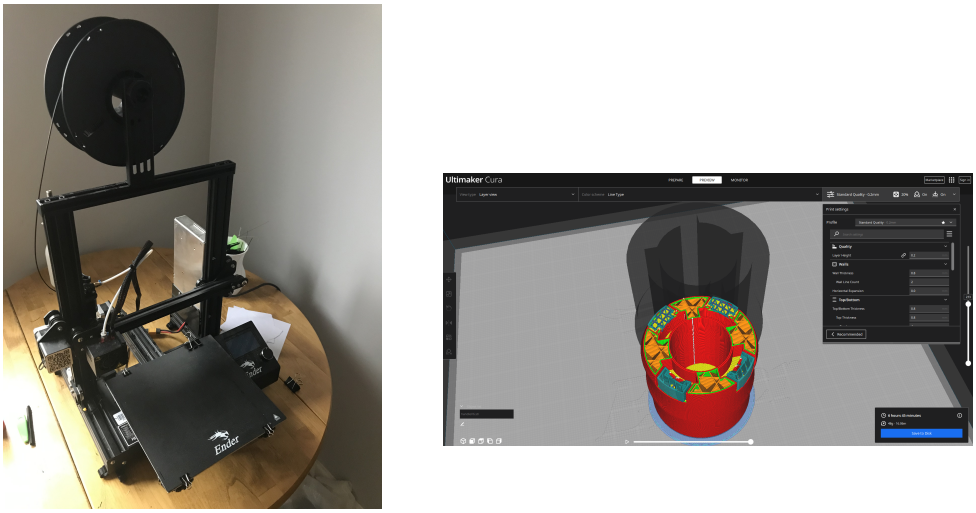

Figure 10: Ender 3 Pro 3D Printer and the Cura environment

2.4 Cloud Data Storage Back-end

The RollSmart will use cloud-based storage to hold data collected about the rollator and allow the user of the RollSmart, authorized medical professionals, or another supervisor to view the data from anywhere. The cloud storage solution will be hosted on Google Firebase. This cloud database

solution provides several solutions for user authentication in several methods, which can range from email, phone, Google, Microsoft, Twitter, and many more. Firebase provides many libraries for different programming languages and offers substantial documentation and support. If there are any issues that arise in the database-related area, the extra documentation can be beneficial in saving time and finding solutions quicker. Close to all the user and sensor data is stored on the Firebase real-time database, which is stored in a JSON structure. The JSON structure is beneficial for the RollSmart GUI application as parsing the data is quicker due to JSON's compact structure. As each user data is categorized into separate areas. A sample of the current Firebase real-time Database structure can be seen in Figure 11 This reasoning, along with Group 33 members having prior experience with Firebase, means we are able to streamline the cloud storage portion of the RollSmart.

The workflow behind the process of using Firebase as a host is as follows. The Raspberry Pi located on the rollator will upload collected data once a day when the device is put on charge. When a user wants to view the data, they will launch a Python application that will pull new data from the cloud database. The system will require an email and password login for anyone attempting to view user data via the Firebase Authentication method. The database will have tables dedicated to each rollator and individual sensor, and all entries will have an associated time stamp.

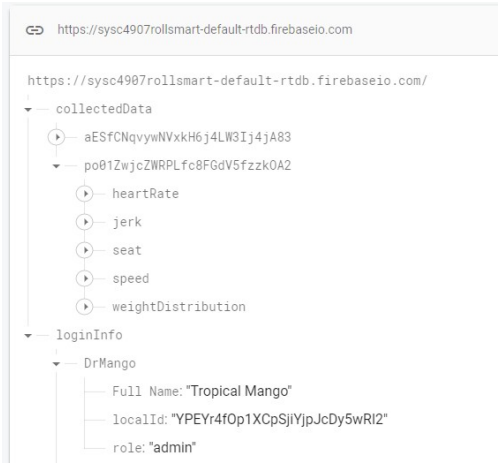

Figure 11: Proposed Firebase Organization Structure

2.5 Data Analysis Application

The RollSmart comes with a Grahpical User Interface(GUI) software package that a user can access a visually appealing and informative data application. The RollSmart functionality requirements per individual include authentication, a variety of data access protocols per role, a summary of collected data, and its visualization. The software includes different pages like a sign-in page, a user list, user information with a 24-hour summary, and a detailed analytics page. The application structure flow of the RollSmart Desktop Application can be seen in wireframe diagrams provided in Figure 12 below.

The requirements for each of the following pages are as follows. The authentication page requires an email and password. The user list has a list of all RollSmart users, this access is only granted to medical practitioners. The user information page includes the details of the user: name, age, date of birth injury information. Detailed analytics requires the data stored on the cloud, this includes all the logs of the rollator from the various sensors along with the associated timestamp. The admin user enrollment form requires information about the user that will be projected on the user information page.

The application is created on a Python framework. After some brainstorming and research, it is found that the PyQt5 library serves the best for the Data Analysis Application when compared to the popular alternative libraries such as PySide2, Tkinter, and wxPython. The PyQt5 library features a lot of resources available to build a robust front end. However, with its complexity, it did have a major learning curve. One of the major benefits of PyQt5 is the ability to read UI files. The Majority of the front end can be designed on QT Designer software, provided by PyQt5. This allows for a lot of customizability without sacrificing time for designing the front end of the application. All the pages were designed on QT Designer and later in the source code were altered to access/view the data in the application. Overall, PyQt5 serves as a solid choice with its numerous amounts of extensions and functionality.

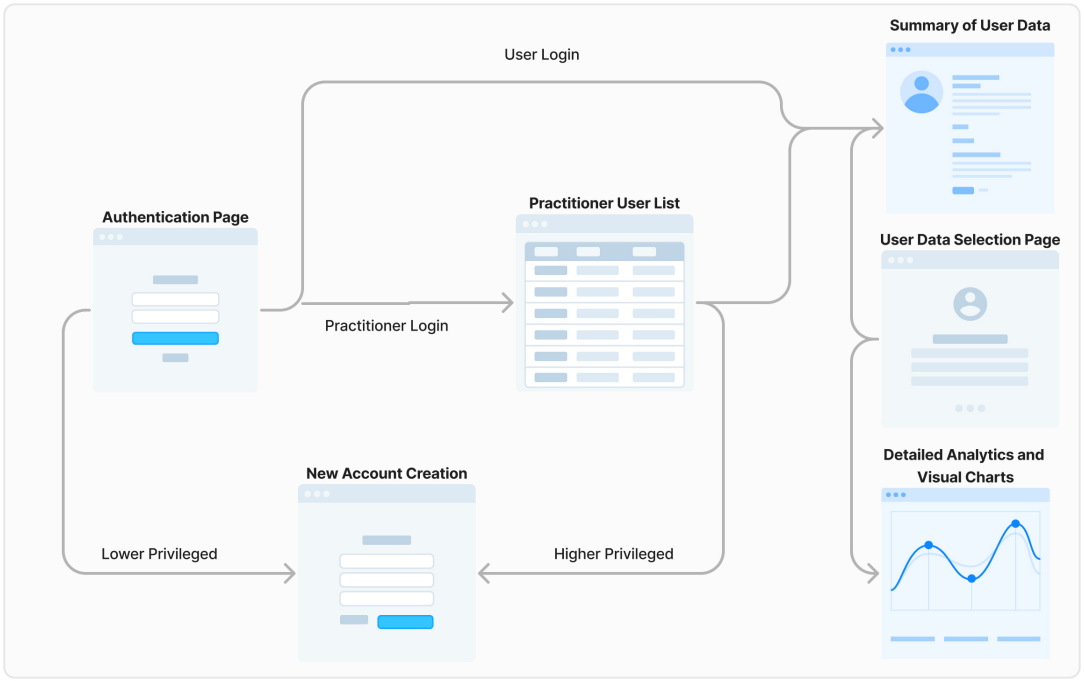

Figure 12: RollSmart Desktop Application Structure

2.5.1 Proposed Application Layout

The proposed flow of the application is as such: The user is greeted with the authentication page like Figure 13 shown below. On this page, they input an email and password. The email and password are verified by the built-in Firebase authentication. Upon being verified, each user and practitioner' will have a unique 28-character local user id linked to their account, this local id will be referred to the user in the entire back end of the code. The identity of the user is checked, if their role indicates to be a practitioner, they will be taken to the user list page like Figure 14 shown below. The practitioner will be asked to choose the user's information they want to view. Upon selection, the user information page will be presented. If it is the user of the rollator, they will be taken directly to their information page like Figure 15 shown below. Now that the practitioner and user are on the same page, they can choose from visualized data of the user's movement throughout the day, the average pressure the user applies on the handle, and their heart rate throughout the day like Figure 16 shown below. These data points can be cross-checked with other data points for the medical practitioner to see if there is any correlation in the user's day-to-day recovery.

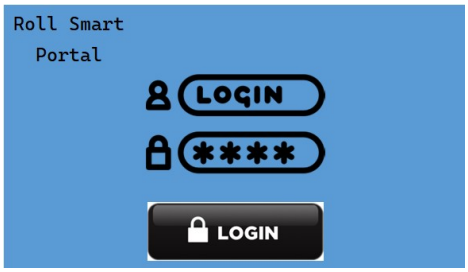

Figure 13: User Authentication Page

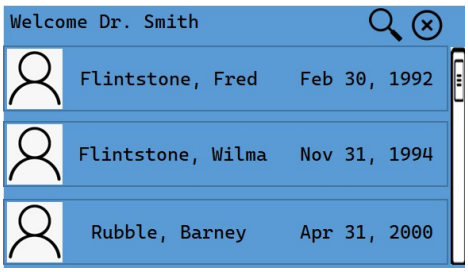

Figure 14: Sample Patient List

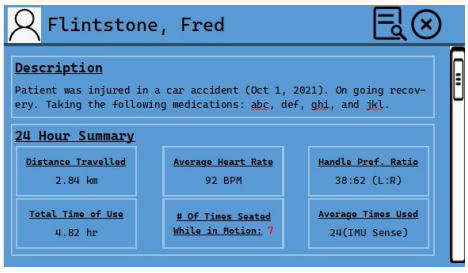

Figure 15: User Information

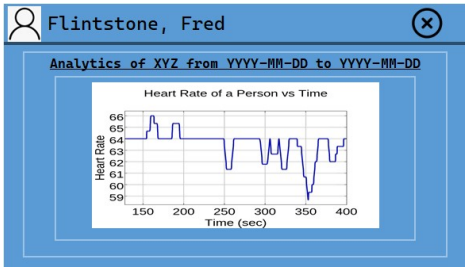

Figure 16: Graphical Representation

2.5.2 Real Time Application Layout

The initial proposed design relatively meets the current application layout with a few additional features that once developed made more sense to implement. The following below will walk through each of the real-time application pages, its layout, the features it encompasses, and its functionality of it.

The **Authentication Page** as shown in Figure 17 requires 2 text inputs and has 2 button presses available. The two text inputs include the person's email and password and the two button press is for creating a new account and submit button. Should a person with an account want to sign in to the RollSmart application, they would need to enter their email and password and press the submit button. From the back end side of the application, it takes the person's entry, and using the Firebase API, it will authenticate the user using Firebase built-in Authentication. As the authentication of the person is done on the Firebase side. The user password and email are not referenced anywhere else in this application. The algorithm that Firebase uses to encrypt the password is SCRYPT, which is a more complex and secure algorithm for these instances. If the user entry is valid, a user's unique local ID(UID) is provided for the user to be referenced throughout the rest of the application. The UID is cross-checked with the users in the loginInfo table in the database to determine their role, if a user is a practitioner, they will be forwarded to the UserList page, and if the role is user, they will be taken directly to their own User Dashboard page. If the user entry is invalid, a popup error message will be prompted, that notifies the user, their credentials are invalid. Should a prospective RollSmart user want to create an account. They are able to do so when pressing upon the create new account button. Which will direct them to the New Account Creation Page as shown in Figure 22.

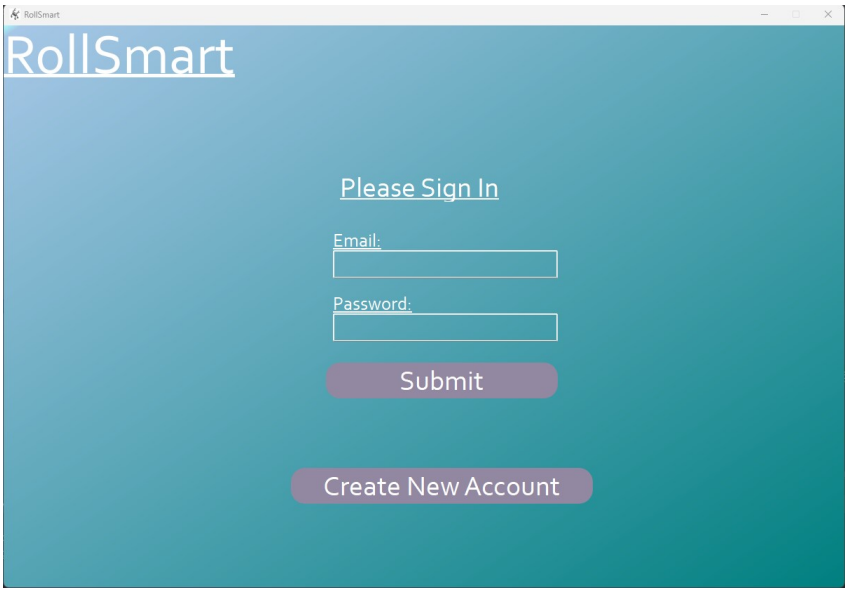

Figure 17: Authentication Page

The **User Selection Page** as shown in Figure 18 is intended for practitioners to select the desired user's data they would like to view. It includes a Greeting for the practitioner at the top, a sign out button, a create new user button, a user list table with the user's general info, a row selection box, and a submit button. The Sign-out button will bring the practitioner back to the authentication page. The create new user button will direct them to the new account creation page. The user list table and row selection box work symmetrically with one another. When a practitioner wants to select a user, they can choose a single box on that table, and the user data is fetched from the row. At the same time that happens the row selection box is updated with the associated row. Alternatively, the user does not have to interact with the table, and can simply select from the row selection box the user's data they would like to view. Upon the submit button being pressed after selection, the associated user dashboard page is loaded as shown in Figure 19.

Figure 18: Practitioners List for User Selection

The **User Dashboard Page** as shown in Figure 19 is the dashboard that the user will see upon signing in and a practitioner will see when selecting the associated user. The user Dashboard includes the user's name at the top, their description followed after, a 24-hour summary table, a detailed analytics button, along with a return and sign-out button. The return button is only visible to a practitioner when a user is signed in. The button is otherwise not visible to the user, as there is no previous page for them to go to besides the authentication page. The 24-hour summary page features some of the users' statistics on their usage patterns of the rollator. The final iteration only includes 3 readings, which are the average heart rate of the user, the average speed they travelled, and their average SpO2 readings all for the past 24 hours. This page is intended to be an intermediary page, if a user would like a quick summary of their stats. Should a user want to see the data plotted over a longer series of time, they can navigate to the User Detailed Analytic Selection Page as shown in Figure 20.

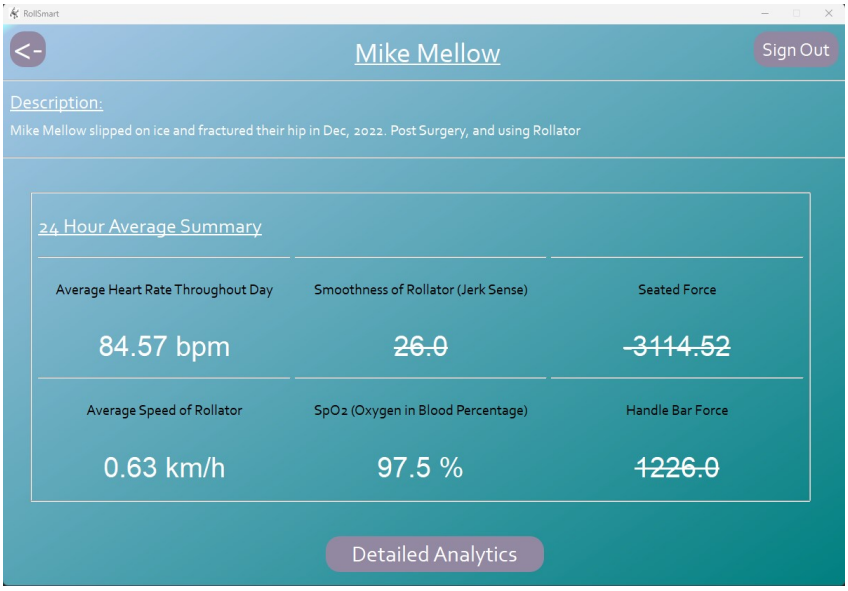

Figure 19: User Dashboard

The **User Detailed Analytic Selection Page** as shown in Figure 20 is where a user and practitioner can select a time frame and view the according data for it. In this page, there is a note section, which provides details on the first and last recorded date, that the user has used the rollator. On the right side of the note section, is where the user can specify the start and end date they want to view. Should a user define a date out of the boundary of the available data, the program will pick the first and last applicable date that is logged for the user. Currently, the application only plots their heart rate data over time. After the user presses submit. The application generates an HTML file with Plotly Graphs rendered on it. A sample output of the graph is shown in Figure 21. The user can interact with the graph and grasp the overall idea of how they're progressing all within the web browser. The HTML file that is generated for the graph is stored inside the application under the CachedGraph folder. When the application is exited or if a user presses sign out, the HTML file is then deleted from the folder.

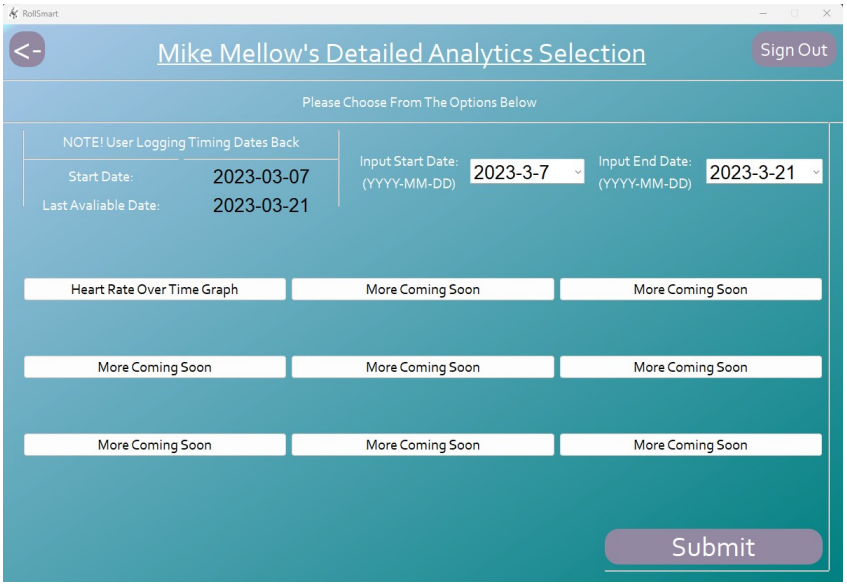

Figure 20: User Detailed Analytic Selection

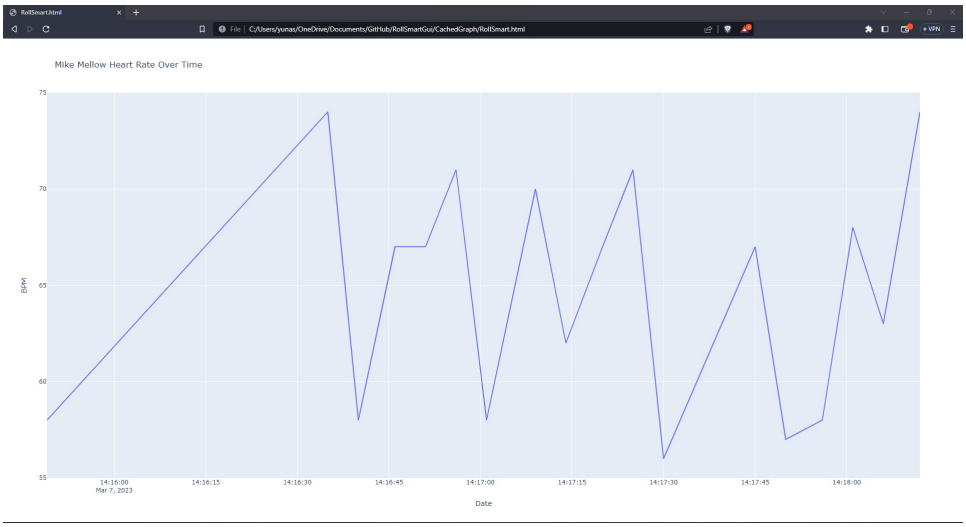

Figure 21: User Graph Output

The **User Creation Page** is shown in Figure 22. There are 3 main categories in the form, which include user detail, account credentials, and type of account. The user detail is stored in the LoginInfo table of the database, these include their name, date of birth and a description. The account credentials that includes the email and password are stored in the Firebase Authentication section, where the UID is generated. The type of account is where the privileged access is set, with the two types being User and Practitioner. When a user creates a RollSmart account on the authentication page, only a user account be created, as the type of account selection box is locked to it. On the UserList page, when a practitioner is creating an account, from the UserList page, this selection box is enabled, and over there both types of selection can be made. Once the form is completed and submitted an indicator at the bottom will display a status message, and all the input fields become locked. Should one need to create another account, the reset button, will clear out the form and enable the fields on the form. On the end-user side, they will receive an email notification for a RollSmart account creation with their credentials disclosed similar to those shown in Figure 23.

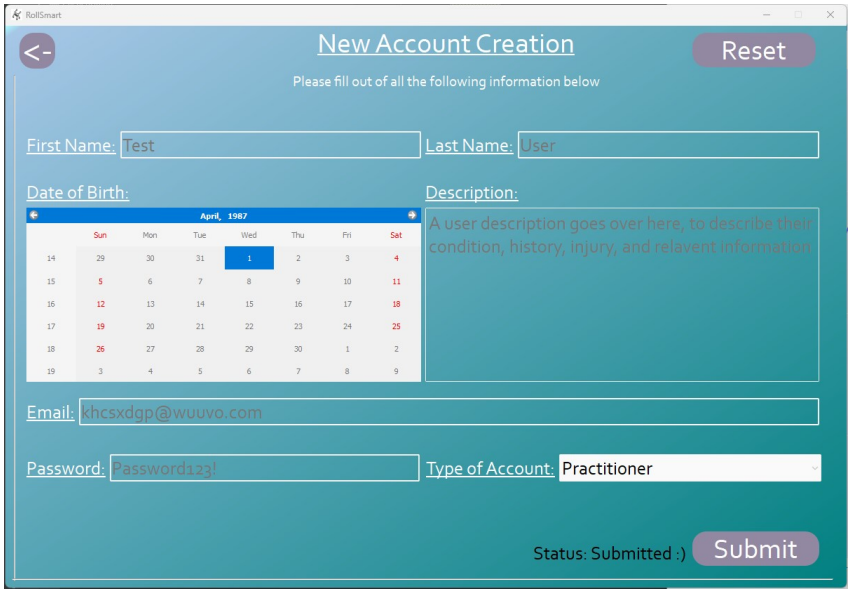

Figure 22: New Account Creation

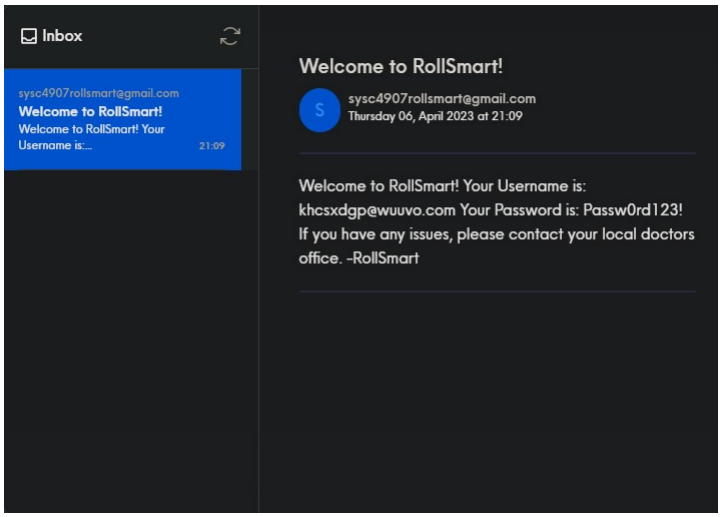

Figure 23: Sample Account Creation Email

With all these moving pieces in the application, the **Process flow** of the RollSmart application can be best summarized in Figure 24.

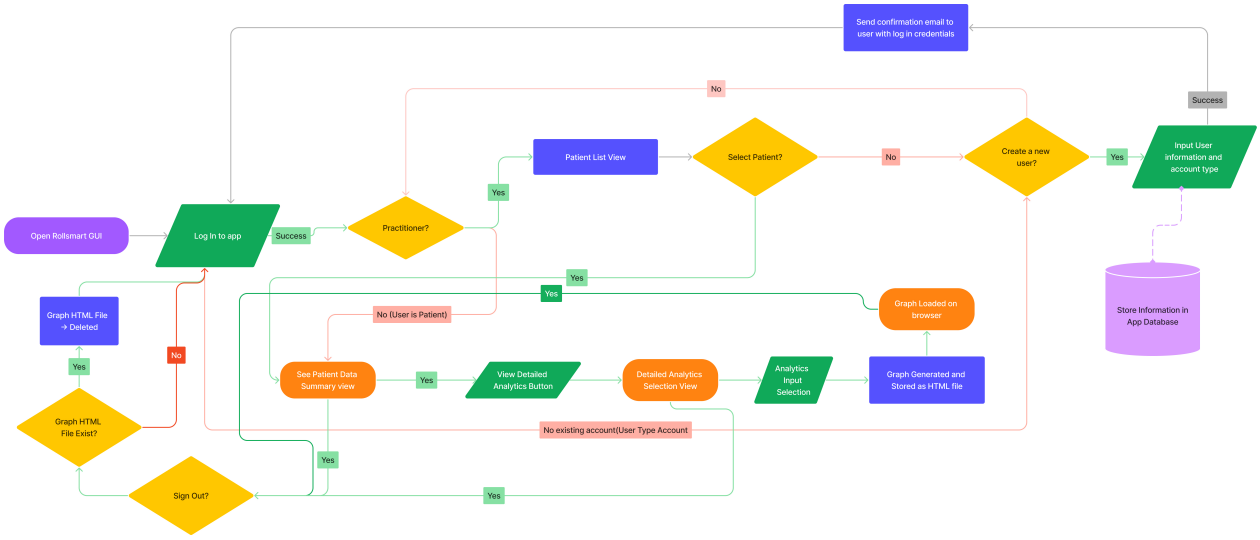

Figure 24: Process flow of Rollsmart Application

2.5.3 Future Implementations For Application

For further future development of the application, a few features to include such as having RollSmart users grouped in sections, adding a mail-to button for bi-directional communication in the app, a pop-up notepad for practitioners to log any specific details of a user for tracking purposes, having the detailed analytic become a more interactive dashboard. For the grouped sections, each person belongs to a layered group. The group would be in a tree structure where, the branches are associated, with a province, hospital/clinic, practitioner, and users within the practitioner group. With this method, only practitioners within a hospital have access to set user profiles. This provides further privacy and makes the Application more scalable for the macro environment. Inside the application, a nice-to-have feature would be is, where a user or practitioner can have a mail-to button, where a user or practitioner can address if there were any points of concern in a chart or in the 24-hour summary, that can be addressed, without needing to phone or exit the application.
Practitioners have a lot of patients, with a lot of patients, includes a lot of files. Having a notepad within the application can streamline the practitioner's workflow. They can log anything that may require future monitoring or something to check back later to see if a user was having a one-off day, or if there are issues with their rehabilitation. The last point of future improvement would be is to have the detailed analytic button on the user dashboard page, generate a full-scale dashboard with multiple graphs on a web page and have further data analysis. Upon research, the Plotly library used for generating the output graphs can also be used to create full-scale interactive dashboards.

3 System Design

3.1 Speed and Distance

Requirements: Compact to be unobtrusive to the user, provide accurate instantaneous speed and have a low power consumption.

The RollSmart should track and monitor the movement speed of the RollSmart. Monitoring and tracking the movement speed of the RollSmart over a period of time to create a trend line will aid a medical professional to determine if the user is progressing positively in rehabilitation through an increase in average and peak movement speed. Speed and distance measured from the RollSmart will be used to track how often the RollSmart is used and the duration of these activities. Tracking usage time and collecting speed data to display to a medical professional will aid in determining rehabilitation progress. Medical rehabilitation devices such as the RollSmart will only deliver desired therapeutic effects if they are used by the user as long as intended. Such a sensor would ideally be unobtrusive to the user while providing accurate speed information.

There are a variety of different sensor technology options available to us that are able to provide these requirements. The sensor would ideally be mounted laterally beside one of the rollator wheels and measure wheel rotations (and thus derive a speed and distance from a known wheel circumference). A reed-style switch and a hall-effect sensor were analyzed and considered. A reed-style switch was chosen over a hall-effect style switch due to reed-style switches closing only when there is a magnetic field present (which is ideal because it consumes no power when opened as opposed to a hall-effect sensor that will stay open when a magnetic field is removed thus draining unnecessary power). The reed switch is an ideal sensor to track usage time as it will be able to detect low-speed movement activities along with high long-term accuracy, whereas sensors such as an IMU (Inertial Measurement Unit) would have trouble with low-speed detection and is susceptible to long term inaccuracies due to IMUs deriving velocity from a measured acceleration.

Magnets are attached to the wheels of the rollator and their turning past the reed switch closes a circuit to produce an output signal. The rate of these signals, the diameter of the wheel, and the time frame the signals are produced determine the speed traveled and distance covered by the user during their RollSmart activity.

We chose the Littelfuse 59025-020 Reed sensor shown in Figure 25 because it is simple, compact, and requires very little power to run. The Littelfuse 59025-020 reed switch provides accurate instantaneous speed information, requires low power compared to similar hall-effect style sensors, and is compact (6.22mm x 25.4mm) so as not to become obtrusive when mounted to the wheel.

Figure 25: Littelfuse 59025-020: Magnetic Reed Switch

3.2 Heart Rate Monitoring

Requirements: Compact to be unobtrusive to the user, provide accurate heart-rate information, and low power consumption.

Data about the heart rate of the user is collected by the RollSmart to aid a medical professional in evaluating rehabilitation progress and overall health. Such a sensor would ideally be unobtrusive to the user while providing accurate heart-rate information.

When your heart beats, capillaries contract and expand. These contractions and expansions alter the volume of blood in the capillaries. The PPG heart-rate sensor emits red and IR LED light to detect and calculate these changes in blood volume and translates these changes into a heart rate. One heart rate sensor will be mounted on each of the RollSmart's handles to monitor the user's heart rate through measurements of the user's fingers/hands. Two sensors are used to provide redundancy in the case one hand does not produce ideal readings, and also allows the system to average both sensors to get the most accurate readings in case one sensor produces jumpy results.

The MAXREFDES117 heart-rate sensor Figure 26 is a reflective PPG optical heart-rate sensor. A reflective-based PPG sensor was chosen over alternatives like ECG due to the need for ECG to have leads attached to the user's body to measure heart rate and transmission-based PPG where sensors are placed on opposite sides of the finger to measure the user's heart rate. A requirement for the RollSmart is to be unobtrusive to the user and the reflective-based PPG sensors only require contact with the skin at any point, so a hand or fingers gripping the handle is a convenient spot for information to be collected. This sensor was chosen because of its low power consumption, small form factor, unobtrusive reflective PPG style design, and its accurate and instantaneous heart-rate output.

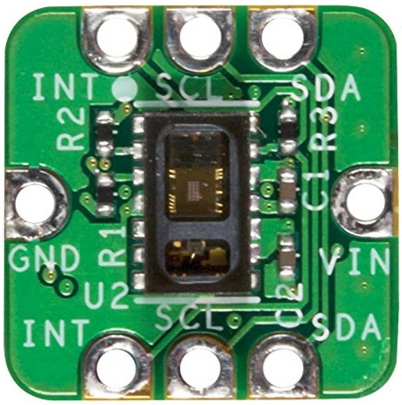

Figure 26: MAXREFDES117: Heart Rate Monitoring Module

3.3 Pace Measuring

Requirements: Unobtrusive to the user, instantaneous acceleration.

The RollSmart will track the fluidity of the user's movement patterns to gauge their pace and determine if the user walks with any sort of limp or jerky motions. The primary data points measured are acceleration, but orientation is also monitored. Drastic changes in acceleration (jerk) will indicate a user is walking in bursts, with a limp, or the rollator has been in a collision. Drastic changes in orientation (ie. the RollSmart is upright and suddenly it is sideways) can also indicate a user falling down. Such a sensor would ideally be non-obstructive to the user while providing accurate acceleration, and orientation.

The Bosch BNO055 IMU sensor fulfills these requirements and since most IMUs are very similar in functionality, we chose this based on its availability to us. It will be used to detect movement, track acceleration, and orientation. The BNO055 IMU sensor shown in Figure 26 consists of multiple integrated sensors: an Accelerometer, Gyroscope, and Magnetometer to function as a 9-axis sensor using I^2C.

During RollSmart activities we expect both linear accelerations, and changes in angular velocity to occur (ie. walking in a straight line and taking turns). The BNO055 can directly measure both of these types of acceleration using a built-in tri-axial 14-bit accelerometer with programmable acceleration ranges ranging from +/- 2g to +/- 16g. During normal operation we only expect a maximum of 1g to be reached. However, during a fall we can expect a large variation in the maximum acceleration reached depending on the environment. Orientation can be easily determined due to the BNO055's onboard sensor fusion.

3.4 Seat Occupancy Detection

Requirement: Able to differentiate between objects placed on the seat by the user (ex. bag of groceries) vs user sitting on the seat, non-binary output, unobtrusive to the user, have a minimum range of 0-30 kg.

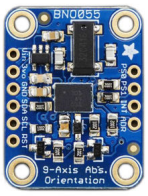

Figure 27: Bosch 9-Axis BNO055 IMU Sensor

The RollSmart will track how long and how often the user is seated on the rollator. This information is used to gauge how much exertion the user is capable of before they need a break, and how long it takes them to recover. Since people also use the seat of the rollator for transporting cargo such as groceries, the system needs to be able to differentiate between the two cases of seat occupancy. This is accomplished with load cells and reading the weight applied to the seat to see if it is over the cargo weight limit, which would mean a person is seated. This load cell along with the BNO055 IMU and reed switch will also determine if the user is misusing the RollSmart by scooting and pushing themselves around while being seated.

A NEXTION WI1802AX4+WI0728 load cell, shown in Figure 28, has been chosen for the task of determining seat occupancy. We explored options such as a binary switch that detects a weight applied, but this would not work as we require differentiation between when the user places items (such as a bag of groceries) on the seat vs when the user sits on the seat. We can assume that groceries will not exceed 30 kg and treat any weight measured over 30 kg to be a user sitting and any weight measured under 30 kg to be groceries/a foreign object and not track that time as a user's seated time. The NEXTION WI1802AX4+WI0728 load cell sensor has a maximum measurement value of 50 kg, much greater than the 30 kg required.

Figure 28: NEXTION WI1802AX4+WI0728: Load Cell

3.5 Handlebar Strain Sensor

Requirement: Unobtrusive to the user, accurate strain output, robust (not susceptible to variations in user hand placement).

The strain sensors will be used to determine if the RollSmart user has an equal left/right weight distribution when using the device. Measuring the strain exerted onto the RollSmart's handle tube frame over a period of time to create a trend line will aid a medical professional to determine if the user is progressing positively in rehabilitation through a decrease in strain exerted onto the handle

grips. This data can be compared over time for interesting clinical applications, such as monitoring the progression of a user over time to determine the efficiency of current treatment/rehabilitation protocols.

Daoki BF350-3AA strain gauges, shown in Figure 29, have been chosen for the task of measuring handlebar strain. One sensor will be placed on the tube frame of each handle of the RollSmart. The user's left/right balance can be determined by comparing the strain recorded at the left and right handle sensors. The Daoki BF350-3AA strain gauges were chosen due to their small size (able to get precise placement for the most accurate readings). Other force-based sensors also require precise hand placement on the handles to accurately read applied force, but strain gauges measure the strain in the tube frame so any hand placement will still produce accurate readings.

We have updated this component from the flex sensor shown in the proposal. This is because the flex sensor would not have been sensitive enough for the small movements expected from the handlebar tube frame. Measuring the strain in the frame

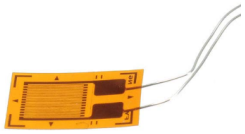

Figure 29: Daoki BF350-3AA strain gauge

A diagram of how the strain sensor and heart-rate monitor will be placed on the handle of the rollator is shown in Figure 30. It will be important to consider the placement of the sensors that maximize the points of contact with the RollSmart user in order to get accurate data.

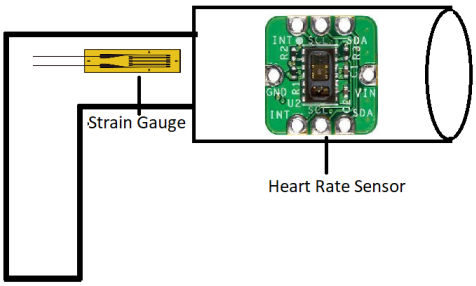

Figure 30: Diagram of RollSmart handlebar Sensor placement

4 Implementation

4.1 Implementation Plan

The RollSmart project consists of several pieces of hardware and software components working together in one final product. With that said, group 33 is using JIRA software to assign, track and manage work that is required to be done. Some of the methods being used to implement RollSmart include the use of a road map, sprints, backlogs, and more. The steps to design the RollSmart include, configuring the microcontroller, reading from sensors, mounting all the hardware, cloud back-end configuration, and building a data analysis application. These are the software and hardware steps toward implementing RollSmart. The SYSC4907 course deliverables such as the proposal, progress report, presentation, and final report are also detailed in a JIRA road map. This ensures that both the project is on track, and requirements are being met and updated. The expanded plan for implementation can be further seen in the designated sections of milestones and risk analysis.

4.2 Strategy

The strategy for implementation of the RollSmart design aims to break down the requirements into simple, progressive steps which will allow us to quickly get a working model of the RollSmart which we can then improve upon. Given the timeline of the project, it is imperative that the Milestones and tasks are clearly defined and easily executable. Recording these tasks and milestones in Jira will allow us to easily reflect and keep track of our progress and ensure we stay on track with our timeline.

4.3 Completed Implementation

- Sensors for each requirement have been selected
- All sensors have been ordered
- GitHub repository has been set up
- Raspberry Pi Zero hardware set up
- Google Firebase database structure established
- Login credentials and authentication have been implemented in Firebase Database
- 3D model of rollator is designed and has been 3D printed for RollSmart test fixture
- Heart Rate, Flex Sensor, and I2C sensor protocol tested
- Reed Switch, Load Cell, Strain Gauge, IMU, and Heart Rate sensors all tested and verified data
- Reed Switch, Load Cell, Strain Gauge, IMU, and Heart Rate sensor Firebase data schema configured
- Local and cloud data storage configured
- All sensors correctly captured and pushed to local and cloud storage
- Testing and validation of all hardware and software completed

Figure 31 visualizes the status of all tasks cataloged in our Jira. As we progressed through the proposal design phase of the project, multiple tasks were added to reflect the changes to the implementation plan.

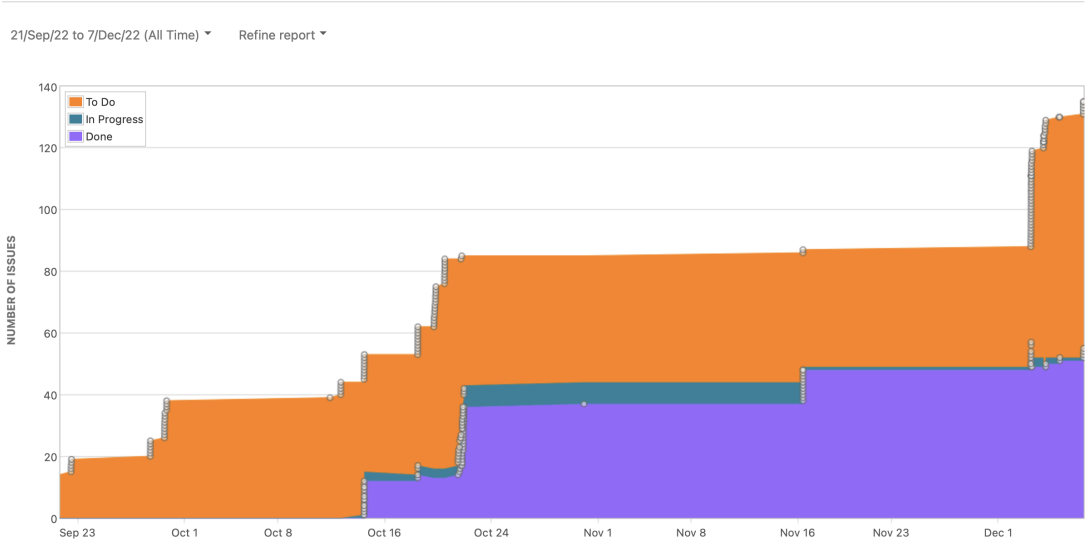

Figure 31: Status of project tasks over time

Figure 32 displays the current overall status of the completion of project tasks as of February 27, 2023.

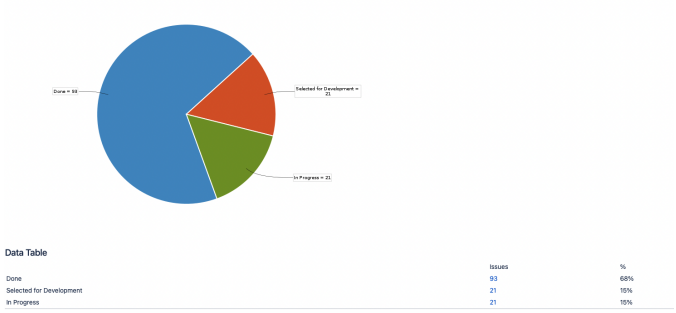

Figure 32: Status of project tasks as of Feb 27 2023

4.4 Data Acquisition

Data acquisition from the sensors on the RollSmart is orchestrated by a main Python script which is called upon the startup of the RollSmart. This script interfaces and initializes connections to the Database and Sensors using their respective Python scripts. Attention was placed on creating clear and concise code for all the sensors, databases, and main modules. In order to create a product that met Industry coding standards and best practices. Specifically, the code conforms to Pylint standards and was continuously tested using unit testing as described in the Testing section of the report. Figure 33 shows a graphical depiction of the Python scripts utilized for the data acquisition from sensors and storage of data using the database script.

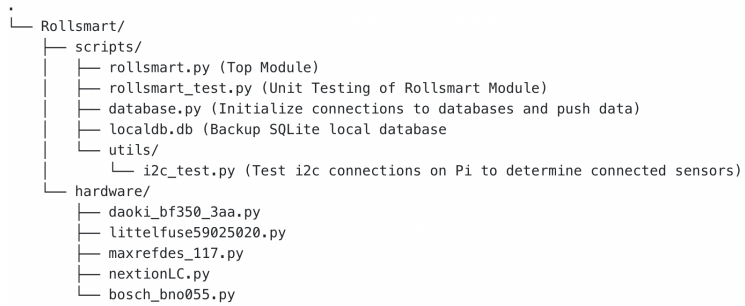

Figure 33: File tree of RollSmart scripts

A Python script for each sensor mounted on the Rollsmart has been created, each of these outlining a class to interface with the sensor. Each script has a 'get-processed-sensor-data()' function, allowing a standardized way to interface with all of the sensors on the RollSmart, while utilizing existing Python modules created by the sensor manufacturers, ensuring the data collection is efficient.

Rollsmart.py is the main Python script that is run upon the startup of the Rollsmart. The flow of this process is shown in Figure 34, which highlights the distinct steps required to initialize the Rollsmart, namely initializing the i2c bus on the pi, initializing each of the connected sensors on the Rollsmart by creating an object of each sensor type, initializing the database object which provides functions to push each sensor data type. Once the RollSmart is initialized, the sensors will be polled sequentially to determine if the RollSmart is in use. If the data indicates that a RollSmart activity is in progress (ie. The data is valid, and indicates a user is currently using the RollSmart, since there is one or more sensors returning valid data), the data is pushed to the database along with a timestamp of the measurement. This process of polling the sensors and pushing the data to the database will keep occurring while the RollSmart is still in use. If the RollSmart has been inactive for more than 15 minutes, the WittyPi will turn the Raspberry Pi off to conserve battery.

4.4.1 Pushing sensor data to database

In order to perform data analysis and visualization of the recorded sensor data for a RollSmart user, it must be stored in the Firebase cloud database. To simplify and standardize the pushing of sensor data, a database ('database.py') script was created. This script describes a Database class, which when initialized, attempts to initialize a connection to the Firebase and SQLite local database. This database module is utilized in rollsmart.py and the main GUI script, abstracting the database

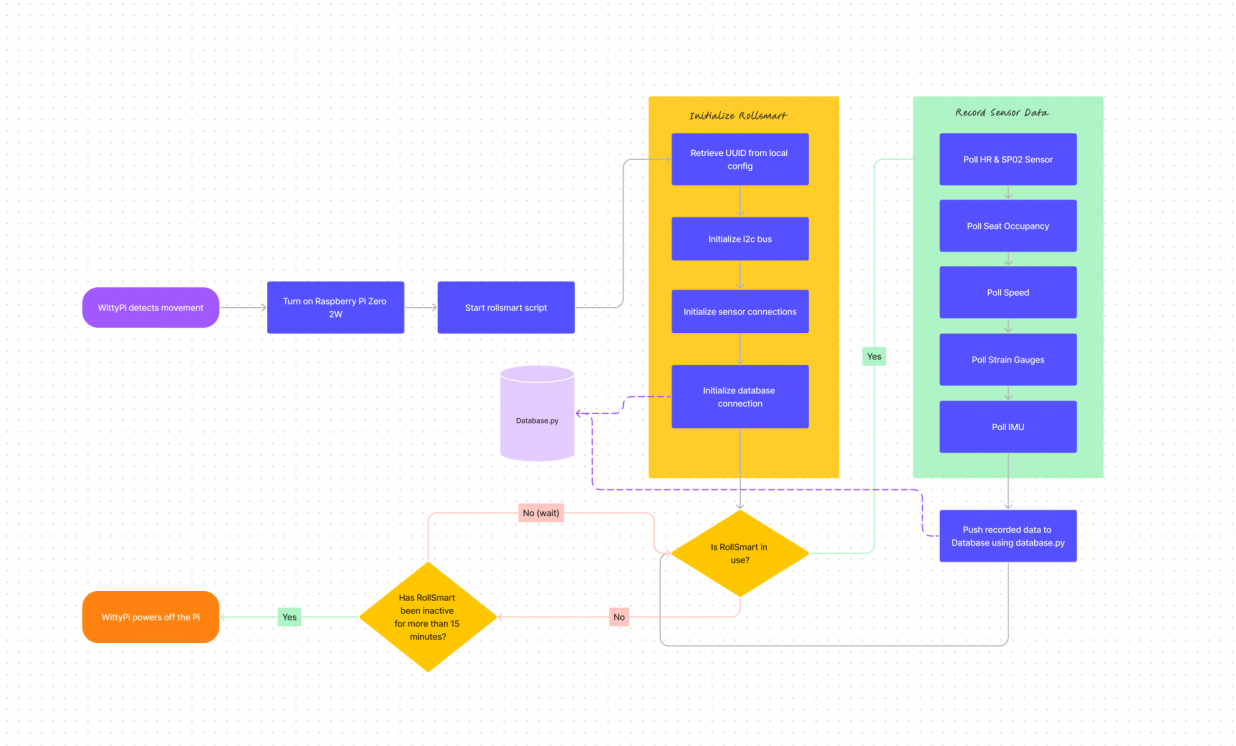

Figure 34: Process flow for Initializing Rollsmart activity and collecting data from sensors

connection interface and limiting reused code throughout the project. The script has a push-data function, which is a wrapper for the 5 sensor-specific data upload functions. A function wrapper was used to offer sensor-specific functions to push data and limit duplicate code which would be utilized by all these sensor-specific functions. Figure 35 shows the Python implementation which allows the sensor data to be stored. This function follows the process flow outlined in Figure 36, storing data locally when the internet connection is not available. If an internet connection is available, data will be pushed directly to the Firebase database and will also check if any backlog data is stored in SQLite local database, pushing this data to Firebase and removing the data from local storage.

4.4.2 Sensor Progress Update

We have ordered all the sensors required for the project, and are currently waiting for their arrival. In the meantime, we have acquired a couple of sensors that are similar but not exactly the version we are using on the RollSmart and started to read values from these sensors with a Raspberry Pi Figure 37. This has allowed us to start figuring out ways to collect data and stream it to a database.

A FlexiForce handlebar sensor was successfully connected to a Raspberry Pi and data was correctly read. Since we have not yet received any Analog to Digital converters, the sensor was connected to an Arduino's Analog Input ports and then relayed via a digital output to the Raspberry Pi. Daisy-chaining the Arduino and Raspberry Pi was only required since we don't have any ADCs on hand currently. Figure 38 shows the circuit used to connect the FlexiForce sensor to the analog

```python
def push_data(sensor):
    """
    Wrapper function which establishes which database is available for data pushing.
    If internet is available, data will be pushed using Firebase ortherwise, data is stored
    locally using local SQLite db.

    When internet connection is available, and there is sensor data in local database, data
    will get pushed

    Args:
        sensor: sensor database key
    """
    def push_data_wrapper(func):
        @functools.wraps(func)
        def wrapper(self, *args, **kwargs):
            try:
                self.check_internet_connection()
                # push new sensor data
                date = args[1].strftime('%Y-%m-%d')
                time = args[1].strftime('%H:%M:%S')
                self.push_firebase(sensor=sensor, uuid=args[0], date=date,
                                   time=time, value=args[3])

                # check for data backlog in local database
                backlog_data = self.sqlite_cursor.execute("SELECT COUNT(*) FROM sensor_data")
                if backlog_data.fetchone()[0] > 0:
                    self.logger.info(f'[bold]DB (Firebase)[/]: Found backlog {sensor} data')
                    data = backlog_data.fetchall()
                    self.sqlite_cursor.execute("DELETE FROM sensor_data")
                    self.sqlite_conn.commit()
                    for d in data:
                        self.push_firebase(sensor=d[0], uuid=args[0], date=d[3],
                                           time=d[2], value=d[1])
            except ConnectionError:
                self.push_sqlite(sensor, args[0], date, time, args[3])

            return func(self, *args, **kwargs)
        return wrapper
    return push_data_wrapper
```

Figure 35: Python code snippet of push-data function

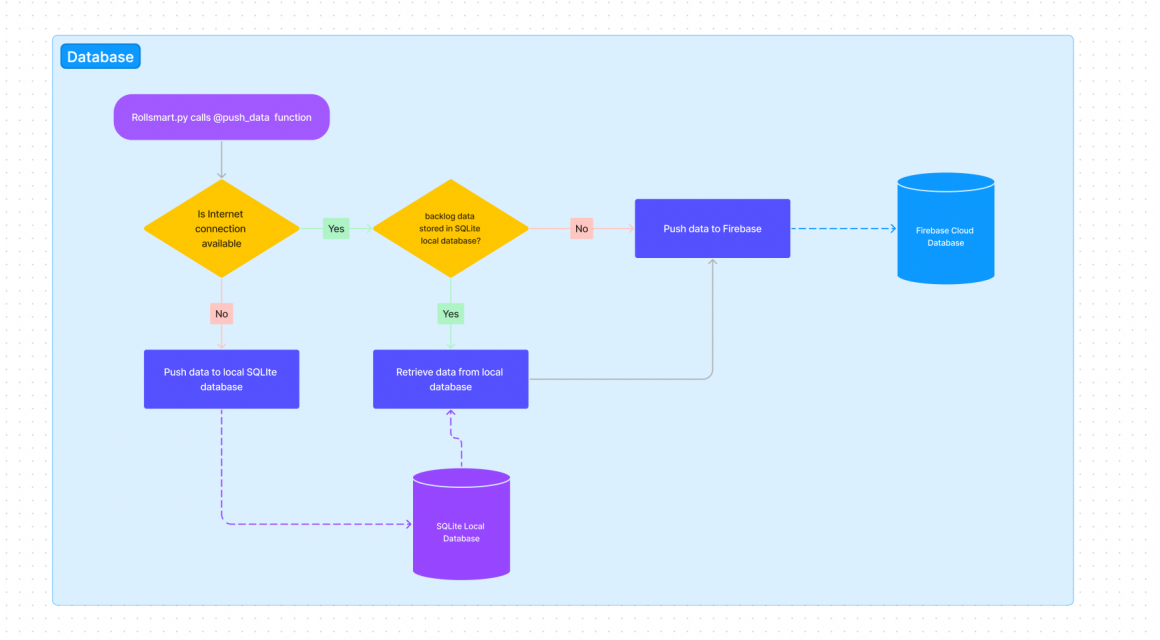

Figure 36: Process flow for sensor data upload to Database

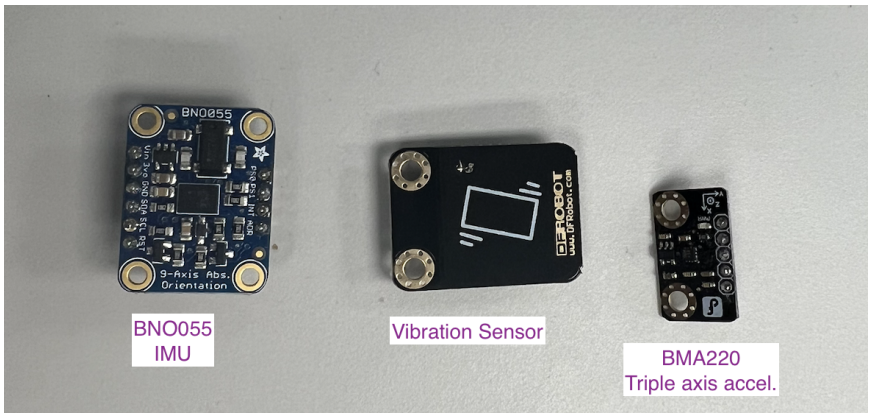

Figure 37: Procured sensors

Input pins of the Arduino.

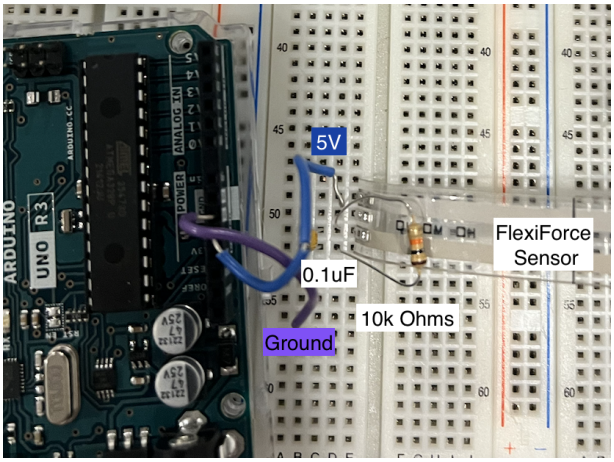

Figure 38: FlexiForce Sensor implementation

Various issues occurred during the sensor hardware and software development. In terms of physical delivery of hardware, there were many shipping and supply chain issues that plagued us. Sensor hardware and miscellaneous components were heavily delayed until late January. Due to these delays, development using alternative sensors were completed but development on the specific final hardware and sensor selected for the RollSmart began in late January to port over work completed on the alternative sensors and hardware. Initially threading was utilized to allow custom and different polling rates for all the various sensors. As development continued, we discovered that by using threading it created an issue for the I2C bus connected sensors to communicate properly. We expect this issue arose with the I2C bus and threading due to mismatched I2C timing caused by our threading implementation. This problem was overcome by utilizing a shared I2C bus with all I2C connected sensors and by utilizing threading exclusively for separate communication protocols to ensure proper timing.

4.4.3 Final Implementation and Installation of Sensors on RollSmart

All of the sensors detailed above are mounted to the RollSmart and hard-wired into the GPIO of the Raspberry Pi. A sample console log taken during a RollSmart activity can be seen in Figure 39. The overall general configuration of all hardware can be seen in Figure 40. An additional closeup photo of the Raspberry Pi along with various sensors and wiring of the RollSmart can also be seen in Figure 41.

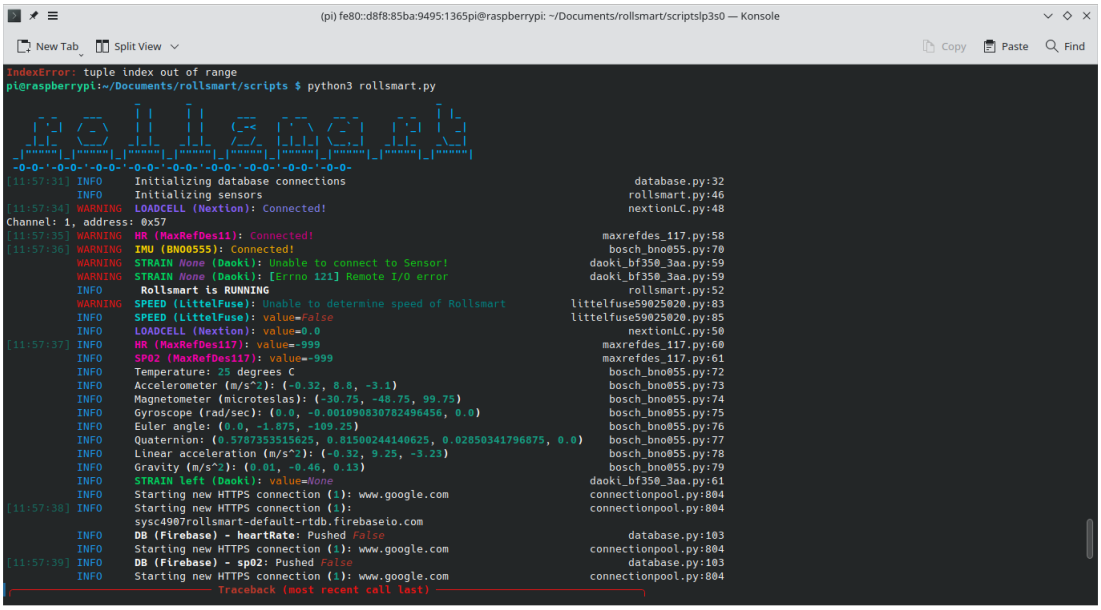

Figure 39: Hardware Sensor Sampling Console Logging

Figure 42 shows the final implementation of all the sensors and their connections to the Raspberry Pi microprocessor. This implementation varies slightly from the initial wiring diagrams which were created during the project planning phase of the project shown in Figure Figure 6.

Figure 43 displays the final physical wiring configuration from the Raspberry Pi to the various sensors as it exists on the RollSmart.

The Reed switch is mounted on the topside of the rollator wheel fork as shown in Figure 44 and connected to the microcontroller through analog wire pairs, with readings transmitted as 500Hz current modulations induced by magnets embedded in the wheels for each fork. The interaction between the Reed switch and magnets can be seen in Figure 45.

The Heart Rate Monitor (PPG) transmits data over an analog output that is communicated via Inter-Integrated-Circuit (I2C) to the Raspberry Pi. There are two heart rate monitors mounted and integrated into each of the Rollsmart's handlebars. The location of the handlebar housing where

Figure 40: Final Rollsmart Product

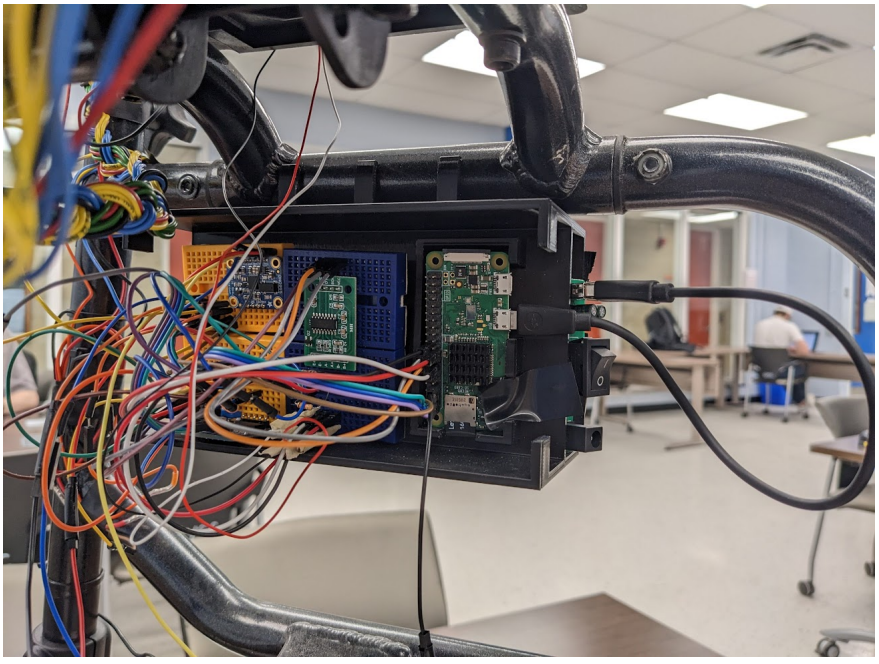

Figure 41: Rollsmart Hardware Configuration

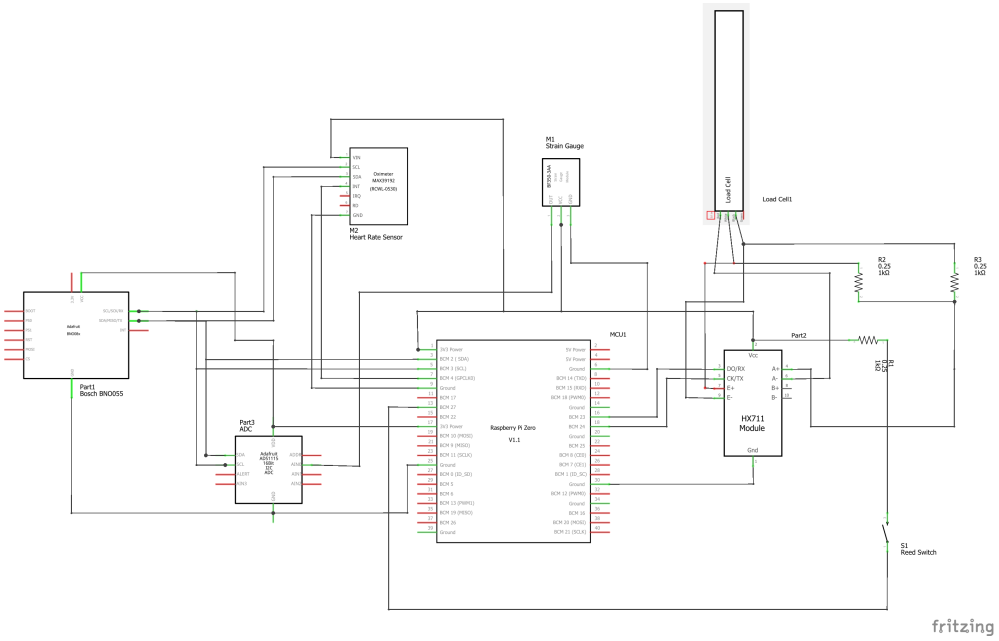

Figure 42: Final Wiring diagram for completed implementation of RollSmart

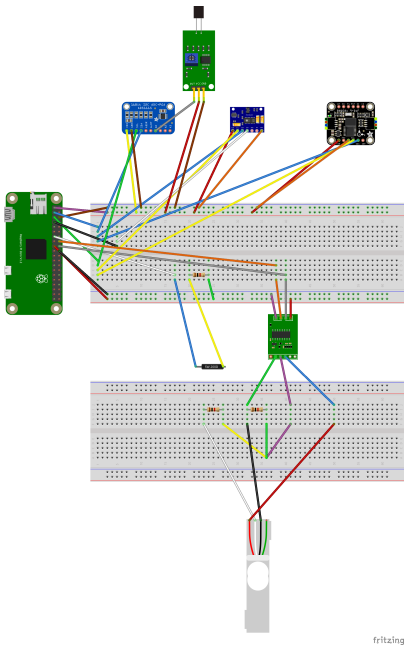

Figure 43: Final Physical Wiring Configuration

the heart rate monitors are integrated into can be seen in Figure 46. These two heart rate monitors integrated into each of the handlebars can be seen in Figure 47 and 48.

Figure 44: Reed Switch Sensor Location

Figure 45: Reed Switch Magnet Interaction

Figure 46: Heart Rate Monitor Handlebar Location

The Inertial Measurement Unit (IMU) supports both I2C and Serial-Peripheral Interface (SPI) communication, of which we use I2C due to its development simplicity and ease of integration with our chosen microprocessor. To ensure accurate data readings, the IMU is mounted near the center of gravity of the rollator housed alongside the microprocessor and wiring as shown in Figure 49.

The handlebar strain gauges as seen in Figure 50 transmits readings as voltage differences between analog wire pairs, each connected to the GPIO pins of the microprocessor via Analog-to-Digital Converters (ADCs) in series with manufacturer-recommended LM358/LM324 operational amplifiers to translate the voltage input into a machine-readable format and reduce error from the source impedance of the strain gauge as a voltage divider.

The load cell is mounted beneath the seat of the Rollsmart as shown in Figure 51 and measures a force applied to the seat. The load cell is a bridge sensor that varies the output voltage given a resistance change in the sensor. The load cell is connected to an HX711 amplifier that acts an Analog-to-Digital Converter to allow the Raspberry Pi to interface directly with a bridge-sensor like the load cell.

Figure 47: Top Mounted Heart Rate Monitor on Handlebar

Figure 48: Bottom Mounted Heart Rate Monitor on Handlebar

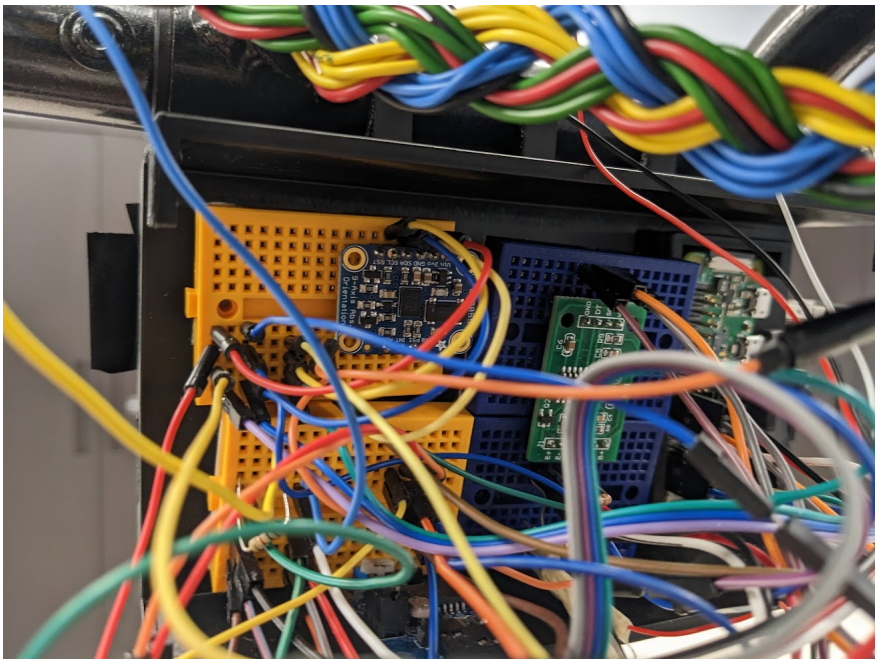

Figure 49: IMU Central Mount Location

Figure 50: Strain Gauge Mounted on Handlebar

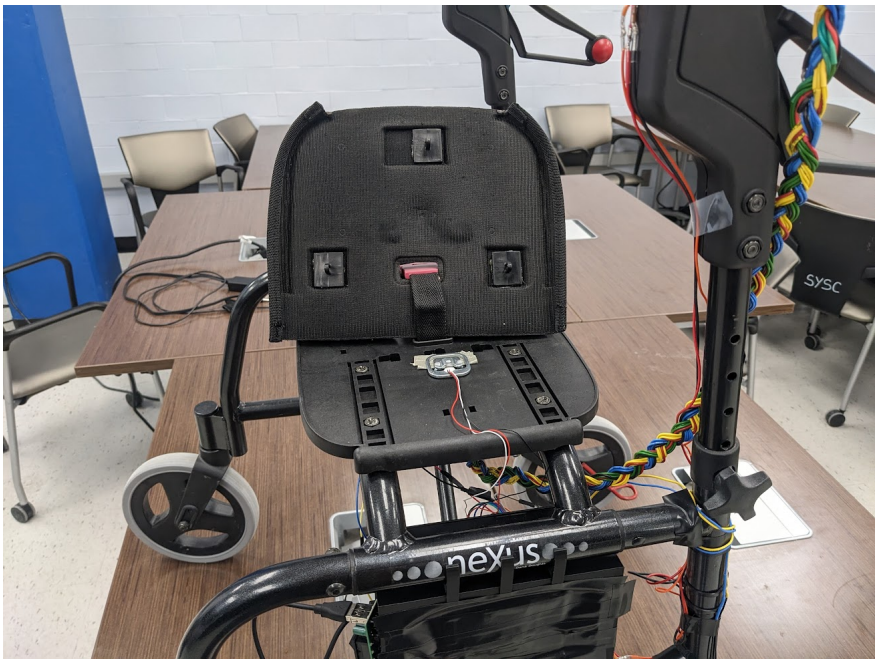

Figure 51: Load Cell Mounted Beneath Seat

4.5 Testing

4.5.1 Unit Testing

To achieve all the functional and non-functional requirements set out for the RollSmart, it is imperative that each component of the project is continuously tested during every stage. Unit Tests have been implemented for each Python script and integrated into GitHub CI in order to continuously test the project's code base. Unit tests allow automated testing of individual modules and units in our programs, ensuring they meet the code design and project requirements.

The unit tests will verify the functioning of the following:

- Successfully connect to Raspberry Pi on the test fixture
- Successfully Connect to sensors on RollSmart test fixture
- Successfully read data from each sensor on the RollSmart test fixture
- Uploading Test Data to Firebase using Python Script
- Attempt to upload to Firebase with incorrect authentication (expected failure)
- Generate User Summary from test Data and ensure it matches expected Summary
- Authentication with valid and invalid user credentials

4.5.2 Sample data for data analysis testing

In order to test the Data Processing portion of the project, test data was generated from each sensor, and used as input to the database upload sequence. This test data is utilized in the rollsmart-test.py unit test module, allowing test data to be pushed using each sensor push-data function, verifying the data is properly uploaded to the Firebase database for data analysis ad visualization.

5 System integration

5.1 3D Printed Sensor Mounting Hardware

To bring all of our sensors and hardware together and attach them to the rollator frame, 3D-printed pieces were made to custom-fit each piece. In Figure 52, the evolution of the piece made to attach the heart rate sensor to the rollator's handle can be seen along with the reed switch mounting hardware progression. These different versions are each created and test fit with sensors onto the frame to find any alterations that need to be made. Some key considerations for the design are the shape of the grips so that it fits comfortably in the hand, the opening for the sensor that allows proper contact with the skin, and the connection point to the frame.

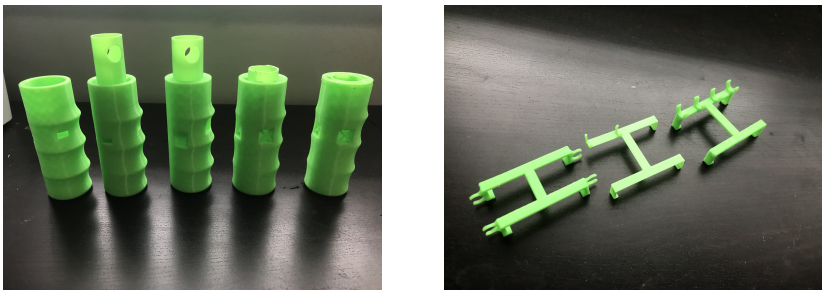

Figure 52: Heart Rate Sensor Handle and Reed Switch Mounts

For the reed switch, the placement relative to the spinning wheel and magnets is important to get accurate readings, so being able to quickly try different configurations was very useful. The heart rate sensor's placement is also tricky because, without proper contact with the skin, the sensor is unable to produce useful readings. Printing many versions with slight changes in placement allowed us to find reliable mounting locations for both sensors. Figure 53 shows the positioning for how the sensors are mounted to the rollator.

Figure 54 shows the battery bank and microprocessor housing that protects the components. Like all of the other pieces, the housing is made to quickly attach to the frame with no tools required. This simplifies the process of converting a standard rollator frame into a smart rollator and allows for some customization in the placement of the battery and microprocessor.

Figure 53: Heart Rate Sensor Handle and Reed Switch Mounting locations

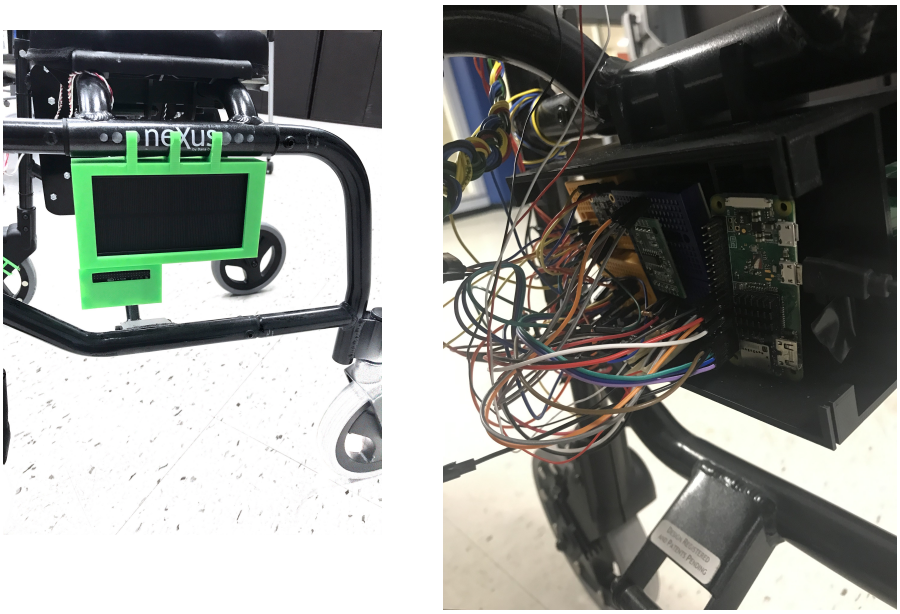

Figure 54: Battery Pack and Raspberry Pi Housing

5.2 Continuous Integration, Delivery, and Deployment (CI/CD)

The project code base is hosted on GitHub, allowing all of the members to work on the project simultaneously with version control. GitHub also supports continuous integration and delivery (CI/CD), allowing automation of building, testing, and integration of code changes in the repository and its deployment. To improve CI efforts, a GitHub Pipeline has been implemented to Pylint all files before they can be merged into the main branch. This means that all code must comply with

PEP8 standards before they can be incorporated into the main code base, ensuring that all the code which will be submitted for this project is robust and of high quality.

This Github pipeline has proved to provide a strict framework for code quality, keeping all team members accountable for creating quality Python code that conforms to industry standards. Another GitHub pipeline was implemented to run the rollsmart-test.py script, allowing any changes to the rollsmart.py main module to be verified as the code changes are committed to the repository.

These CI/CD efforts have proved to be extremely useful in the development of the RollSmart data acquisition solution, allowing the identification of various bugs in the code and the reduction of duplicate code throughout the repository.

5.3 RollSmart Test Fixture and integration into GitHub

A RollSmart unit will reside in the lab and will be equipped with all the RollSmart sensors, it will be constantly connected to power and Ethernet. It will act as a remote test unit, where we can test hardware and software requirements. Ideally, the RollSmart test unit would be integrated into GitHub CI as a worker, running the test sequence at desired intervals and when a merge request is triggered, on the physical hardware. This would allow the code to be continuously verified on the physical hardware with a simulated input, verifying any changes to new code which is getting committed to the code base. Unfortunately, the Raspberry Pi 2W does not have enough processing power to act as a GitHub worker, and given the time constraints of the project, It was not feasible to develop another dedicated test unit utilizing the bigger Raspberry PI 3/4 which could support these operations. Instead, the code was tested on the physical hardware by connecting to the pi using the terminal and executing the test scripts manually.

6 Project Management

6.1 Requirements

The requirements for the RollSmart project are divided into four categories:

1. Abstract Requirements: the descriptions of *what* we intend the system to be capable of based on project group discussions, reviews of the prior art, feedback solicited from Dr. Chan, and restrictions derived from course requirements. These requirements are **not meant to specify implementation details**, but instead, describe what satisfactory system-level requirements will accomplish.

2. System requirements: the requirements for the concrete system design we have created to satisfy the abstract requirements. These can be interpreted as describing ***how* we will accomplish *what* we have decided to accomplish** at a system level. Downstream traceability from the abstract requirements.

3. Software requirements: the implementation requirements of the **required functions and capabilities of the software components of the system** (and their interfaces). Downstream traceability from the system requirements that have been allocated to software components.

4. Hardware requirements: the implementation requirements of the **required functions and capabilities of the hardware components of the system** (and their interfaces). Downstream traceability from the system requirements that have been allocated to hardware components.

The requirements for each of the requirement categories are collected into their own requirements specification tables. Each requirement is labeled by a unique ID, and each requirement row includes a "Downstream" cell that shows traceability to the IDs of the lower-level requirements derived from that requirement row.

At the time of this project update, the component-level software and hardware requirements are still being developed and will not be included in the update report.

6.1.1 Abstract Requirements

ID	Name	Type	Description	Downstream
SR-53	Speed recording	Functional	The RollSmart system SHALL record the speed with which it is moving while in use.	SR-69
SR-54	Distance recording	Functional	The RollSmart system SHALL record the distance it has moved while in use.	SR-74
SR-55	Acceleration recording	Functional	The RollSmart system SHALL record the rate at which it is accelerating while in use.	SR-75
SR-56	Movement duration recording	Functional	The RollSmart system SHALL record the duration of each time span that it is in motion.	SR-74
SR-57	Seating duration recording	Functional	The RollSmart system SHALL record the duration of each time span that the user is seated on it.	SR-78
SR-58	Heart rate recording	Functional	The RollSmart system SHALL record the heart rate of the user while in use.	SR-79
SR-59	Weight application recording	Functional	The RollSmart system SHALL record the strain applied to each handle while in use.	SR-77
SR-60	Minimum charge duration	Functional	The RollSmart system SHALL be usable for a minimum of 24 hours while on battery power.	SR-180
SR-61	Local data storage	Functional	The RollSmart system SHALL store all recorded data locally.	SR-81
SR-62	Local data retention policy	Functional	The RollSmart system SHALL erase all data that has been stored locally for longer than seven days.	
SR-63	Charging mechanism	Functional	The RollSmart system SHALL be capable of wired battery charging.	SR-181
SR-64	Local data cloud upload	Functional	The RollSmart system SHALL upload all local data to the cloud once per day.	
SR-70	Rollator support	Functional	The RollSmart system SHALL be compatible with the Human Care Inc. Nexus rollator.	

Table 4 – continued from previous page

ID	Name	Type	Description	Downstream
SR-71	Battery power	Functional	The RollSmart system SHALL be powered by an internal rechargeable battery.	SR-180
SR-182	Battery power duration	Non-Functional	The RollSmart system battery SHALL be capable of delivering power to the system for 24 hours	SR-177, SR-178, SR-179, SR-180, SR-181

Table 4: Abstract Requirements Specification

6.1.2 System Requirements

ID	Name	Type	Description	Downstream
SR-69	Wheel speed reed switch	Functional	The RollSmart system shall be equipped with a Littelfuse 59025-020 Reed switch sensor configured for detecting and transmitting the speed of the rollator.	
SR-74	Distance measuring	Functional	The RollSmart system shall be equipped with a Littelfuse 59025-020 Reed switch sensor configured for detecting and transmitting the distance traveled by the rollator.	
SR-75	Acceleration sensor	Functional	The RollSmart system shall be equipped with a Bosch BNO055 IMU sensor configured for detecting and transmitting the acceleration of the rollator.	
SR-77	Handle strain sensors	Functional	The RollSmart system SHALL be equipped with Daoki BF350-3AA strain gauge sensors configured for detecting and transmitting the strain applied to each handle individually by the user.	
SR-78	Seat load sensor	Functional	The RollSmart system SHALL be equipped with a Nextion WI1802AX4+WI0728 load cell sensor configured for detecting the load applied to the seat by the user.	
SR-79	Heart rate sensor	Functional	The RollSmart system SHALL be equipped with a MAXREFDES117 PPG sensor configured for detecting and transmitting the user's heart rate.	

Table 5 – continued from previous page

ID	Name	Type	Description	Downstream
SR-81	Controller board	Functional	The SR SHALL be equipped with a Raspberry Pi Zero board capable of interfacing with each of the SR sensors. The board shall be equipped with a real-time hardware clock, a Wi-Fi radio, and a Bluetooth radio.	
SR-177	Power management add-on board	Functional	The RollSmart system SHALL be equipped with a Witty Pi power management board.	
SR-178	Power management add-on board shutdown configuration	Functional	The power management add-on board integrated with the RollSmart system SHALL be configured to completely power off the RollSmart controller board when the rollator has been stationary for five minutes.	
SR-179	Power management add-on board startup configuration	Functional	The power management add-on board integrated with the RollSmart system SHALL be configured to completely power on the RollSmart controller board within one minute of the rollator beginning to move after a controller board shutdown	
SR-180	Internal battery	Functional	The RollSmart system SHALL be equipped with a 36 Ah Lithium Polymer battery pack connected to the RollSmart controller board and power management add-on board.	
SR-181	Battery charging mechanism	Functional	The RollSmart system SHALL be equipped with a Micro USB charging interface connected to the internal battery.	

Table 5: System Requirements Specification

6.2 Justification of suitability

Group 33 is a team composed of members with unique engineering backgrounds and skills well suited to the diverse requirements of the RollSmart project. Corbin, Kenny, and Yunas are Computer Systems Engineering students, Mark is a Software Engineering student, and Isabelle is a Biomedical and Electrical Engineering student.

The Computer Systems degree is helpful in this project because the degree focuses on the ability to integrate software and hardware into a functional system. Since this project will use multiple sensors, a cloud-based data storage system, and software to process data, the system integration

skills learned in the computer systems degree will assist in creating a final product.

Similarly, Software engineering focuses heavily on the methodological aspects of writing software, including requirements engineering, system design documentation, verification and validation, and software quality management; these are essential aspects of engineering and delivering correct and complete software systems.

Finally, a Biomedical and Electrical engineering degree will provide greater insight into the project's biomedical aspects. This medical device utilizes multiple electrical sensors to acquire biomedical data and microprocessors to process and visualize the data.

6.3 Team Member Tasking

Table 4 shows the distribution of the tasks for the project deliverables for each team member of group 33.

Team Member	Task
Corbin	3D printing sensor hardware, database management and meeting planning
Kenny	Sensor testing, validation and script development, and data acquisition
Yunas	Python app development for data display, database management, and meeting notes
Mark	Requirements management, Jira maintenance, and power management
Isabelle	Data acquisition, Sensor Implementation and testing, Database interface script, GitHub maintenance and CI/CD

Table 6: Team Member Tasking for Project Deliverables

6.4 Milestones

The major milestones and task breakdowns for this project are recorded in Jira as detailed in the Risk analysis section of this report. The Milestones required to accomplish the requirements for this project are outlined below.

6.4.1 Configure Raspberry Pi and connect sensors

Getting the Raspberry Pi configured to run an OS and connected to all the sensors we are using is our first step toward building the system. Mark has been assigned the lead role for this because of his familiarity with embedded software development.

6.4.2 Read data from sensors

Once all sensors have been connected to an operational Pi, we will begin gathering data from them. This will mean that drivers for all sensors are working and data can be stored locally pending upload to the cloud. Isabelle has been assigned the lead role for this because of her experience with data collection.

A breakdown of the tasks for these milestones is shown in Figure 55, as recorded in Jira.

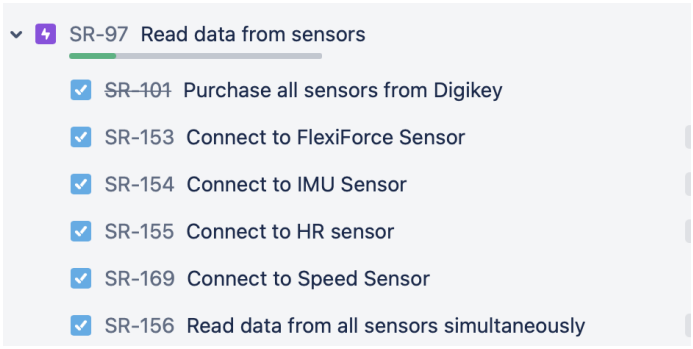

Figure 55: Microprocessor configuration milestones

6.4.3 Sensor mounting

Attaching sensors to the rollator by creating 3D-printed mounting hardware is important because the placement of the strain sensors, heart rate monitor, and Reed switch sensors will impact the accuracy of our data collection. Corbin has been assigned the lead role for this because of his experience with 3D printing.

A breakdown of the tasks for these milestones is shown in Figure 56, as recorded in Jira.

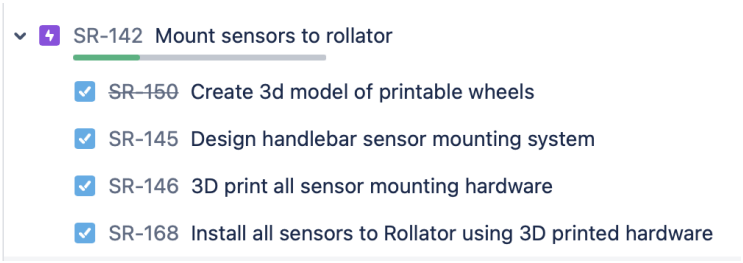

Figure 56: Sensor Mounting Milestones

6.4.4 Sensor data export

Uploading sensor data to the cloud is the next important step in our system integration plan. We will be developing a Python program to pull data from the cloud to create a report for user and doctor viewing. Kenny has been assigned the lead role for this because he would like to become

more familiar with cloud storage systems.

A breakdown of the tasks for these milestones is shown in Figure 57, as recorded in Jira.

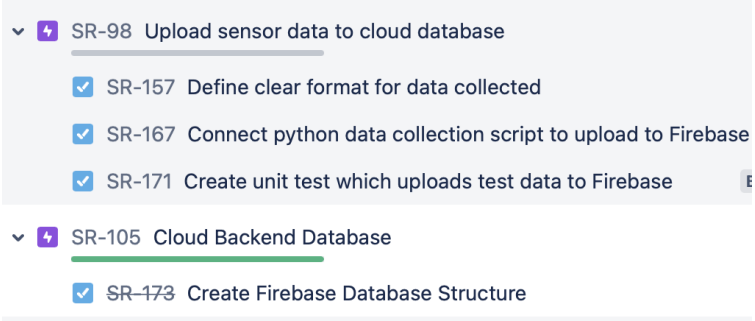

Figure 57: Database Creation Integration to data acquisition Solution

6.4.5 Data Summary generation

Creating a Python program to generate a report of data stored in the cloud database for a medical professional or another supervisor to view is the final step in developing a useful back-end of the RollSmart system. Yunas has been assigned the lead role for this because of his experience with Python.

A breakdown of the tasks for these milestones is shown in Figure 58, as recorded in Jira.

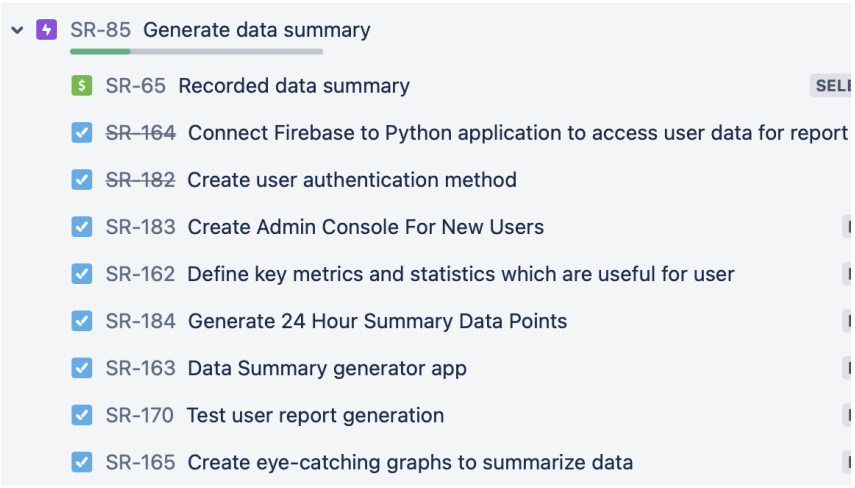

Figure 58: User Data Summary Generation milestones

6.4.6 User test

Having a user test the RollSmart will show that an operational Pi is collecting accurate data, and properly uploading the data to a cloud database, and the Python application creates a summary for review. This will indicate our system is complete.

Figure 59: User Test Milestones

6.5 Project Timeline

The RollSmart project has several major milestones and development stages planned over the course of its development. The Gantt chart in Figure 60 presents the general timeline which we aim to follow to successfully deliver the project.

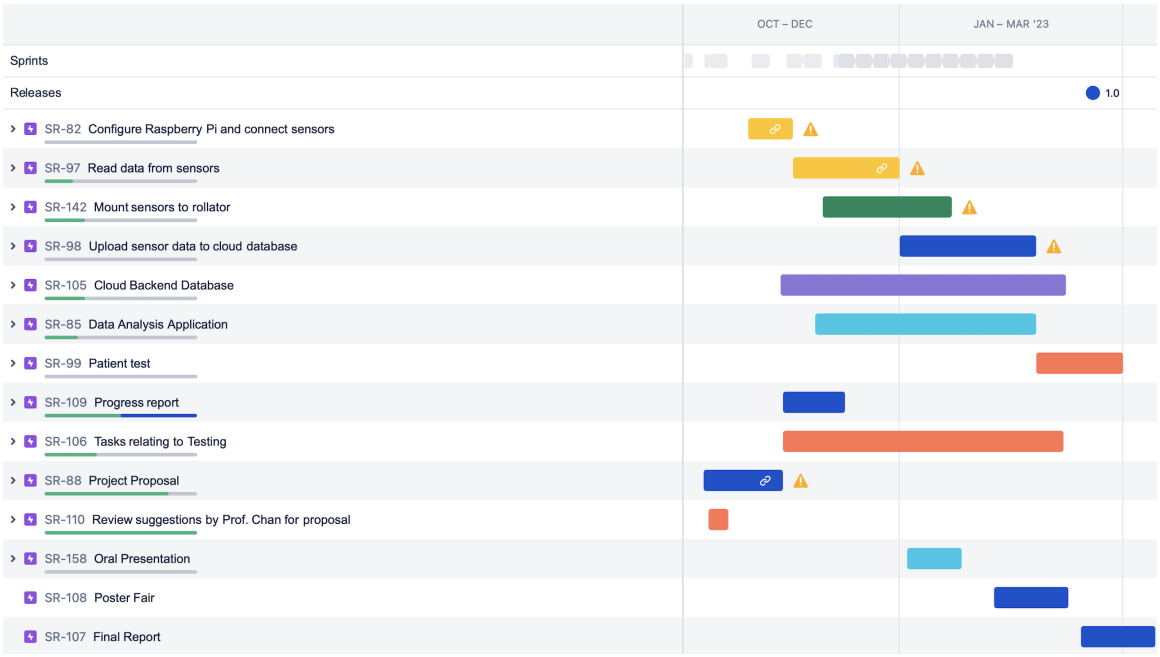

Figure 60: RollSmart Project Timeline

Milestones have been assigned to team members who will act as milestone leads. The lead for each milestone will be responsible for the organization of the work and for distributing tasks to other team members. Another important role of the milestone lead will be to ensure that all the requirements that are outlined in this proposal are being met. Table 7 shows a breakdown of the milestones and their respective team leads.

Milestone Description	Lead	Approximate completion date
Configure Raspberry Pi and connect sensors	Mark	November 18
Read data from wireless/wired sensors	Isabelle	December 31
Mount sensors and required hardware to rollator frame	Corbin	January 20
Export sensor data to cloud storage service	Kenny	February 24
Generate a summary of data for viewing	Yunas	February 24
Have a patient use the rollator to collect data about a short period of activity	N/A	March 31

Table 7: Milestones, Leads, and approximate completion date

6.6 Budget

A bill of materials is outlined in Table 8, listing all the components required to implement the RollSmart design.

Component	Cost Per Unit ($)	Quantity	Total
Raspberry Pi Zero W	65	1	65
Witty Pi Power Module	19.95	1	19.95
Reed Switch	5.39	2	10.78
Heart Rate PPG Sensor	10.75	2	21.5
IMU	47.96	1	47.96
Seat Occupancy Load Cell	12.97	1	12.97
Strain Gauge (Pack of 10)	11.99	1	11.99
Battery Bank	36.76	1	36.76
Analog Digital Converter	8	4	32
Strain Gauge Amplifier	11.23	2	22.46
Total			281.37

Table 8: Bill of Materials for RollSmart

7 Risk Analysis

7.1 Project Delivery

The RollSmart project is vulnerable to a variety of common problems that may impact the delivery of the system, including implementation and testing difficulties from poor requirements, misunderstood allocations of tasks, bugs in the software or the hardware, and interpersonal conflicts between team members.

The probability of exposure to each risk is estimated based on a specific rationale for each risk. The probability of exposure shall be categorized into one of the probability classes, P1, P2, P3, or P4 in accordance with Table 9 as adapted from [1]. The difference in probability between each class is roughly an order of magnitude.

	Class			
	P1	**P2**	**P3**	**P4**
Probability	Very Low	Low	Medium	High

Table 9: Classes of Risk Probability

Similarly, the severity of exposure to each risk is estimated based on a specific rationale for each risk. The severity of exposure shall be categorized into one of the severity classes, S1, S2, S3, or S4 in accordance with Table 10 as adapted from [1]. The difference in probability between each class is roughly an order of magnitude.

	Class			
	S1	**S2**	**S3**	**S4**
Consequences	Minimal	Undesirable	Significant	Catastrophic

Table 10: Classes of Risk Severity

Based on the combined contributions of each risk probability and risk severity, the overall impact to the project of each risk shall be categorized into one of the impact classes, L1, L2, L3, or L4 in accordance with Table 11.

	Class			
	L1	**L2**	**L3**	**L4**
Impact to project	Minimal or none	Noticeable lack of expected work quality	Significant deviation from project plan	Project failure
Breakdown Example	$P[1,2,3] \times S[1]$	$P[1,2,3] \times S[2]$, $P[4] \times S[1]$	$P[1,2] \times S[3]$, $P[4] \times S[2]$	$P[3,4] \times S[3]$, $P[1,2,3,4] \times S[4]$

Table 11: Classes of Risk Impact

A list of potential risks to the project has been developed along with their related mitigations

and residual risks in Table 12, adapted from a study of common project management risks [5] in the software industry.

Risk	**Impact**	**Mitigation**	**Residual Risk**
Project purpose and need are not well understood	L1 ($P1 \times S1$)	Meetings held to clarify the intention of the project before the requirements elicitation stage	Final product may not align with advisor's vision
Project design is incomplete	L2 ($P2 \times S2$)	Scope has been rigorously defined to provide a trivial mapping of requirements to design	An incomplete project design may be submitted
The project schedule is not clearly defined or understood	L3 ($P1 \times S3$)	Jira issues created and assigned for project tasks, with team notifications for deliverable deadlines Regular in-person meetings held to synchronize priorities and maintain awareness of schedule	Deadlines may be incorrectly documented in Jira
Team priorities are poorly managed	L1 ($P1 \times S1$)	Team members are assigned specific tasks that are tracked in Jira	Jira issues may not be created for all tasks
Mistaken estimation or scheduling	L2 ($P2 \times S2$)	The mitigation for project schedule enforcement also applies to this	Jira issues may not be created for all tasks
Ineffective online communication among project team members	L1 ($P3 \times S1$)	Regular in-person meetings held to synchronize priorities	Interpersonal conflicts may develop between team members
Pressure to arbitrarily run tasks in parallel or with shorter durations, increasing the risk of errors	L2 ($P2 \times S2$)	Breakdown of task parallelism agreed upon as reasonable by team consensus at the start of the project	Unvoiced opinions from team members may have been unaccounted for
Unplanned work that must be incorporated into the project schedule	L2 ($P1 \times S2$)	Assumptions that we made about the project expectations are documented Research into prior art for instrumented rollator systems has been conducted to understand potential future concerns	Fully-unknown problems may develop that will require additional work

Table 12 – continued from previous page

Risk	Impact	Mitigation	Residual Risk
Scope Creep in project requirements	L2 ($P2 \times S2$)	Project scope has been comprehensively outlined in the requirements definition and the project proposal, and will be referred to throughout the project to assess any changes against it	As our understanding develops, changes may be inevitable

Table 12: Risk Identification

We claim that the RollSmart project will be able to deliver on time, on scope, and on schedule based on the mitigations we have in place to enforce our project quality standards and project management expectations. However, we have agreed that our overall goal is to produce a functional system capable of being demonstrated at the end of the project timeline, even if the delivered functionality is diminished relative to what we planned to deliver at the start of the project.

7.2 Team Member Personal Safety

The physical assembly of the RollSmart can be considered on a safety spectrum of low to mid-risk. However, safety is more important than convenience. To produce the RollSmart, group 33 members agree to follow these safety protocols whenever there is any fabrication involved. As the rollator requires a few attachments, if there is any use of power tools, personal protection equipment (PPE) shall we worn. An example of this is, if there is any form of drilling, members are expected to be wearing appropriate eye protection, gloves, and suitable clothing coverage. The RollSmart has various sensors equipped that will require some form of soldering to make a strong connection. In this situation, the following must be done to ensure safe operations are executed. The work environment is clean and tidy, PPE is worn anytime the iron is on, there is a damp sponge to absorb excessive solder, a fan to dissipate the solder fumes from the area, the iron is returned to a secure stand, and is turned off when it not in use. Some general considerations are to ensure there is a first aid kit readily available anytime the rollator is being worked on. Dealing with electrical components, it is important to have a first extinguisher nearby, if that is not applicable then the next best option is to be informed of the nearest location of one. The majority of the physical production of the RollSmart will be taking place in the SYSC Capstone room, which will provide a healthy and safe environment for the group 33 members.

7.3 Functional Safety

The Human Care Inc. Nexus rollator model on which the RollSmart system is based is certified as a Class 1 medical device by Health Canada as a low-risk non-invasive system. Given that the existing design of the rollator will not be modified by the addition of sensors, we assert that no additional hazards related to the functional behavior of the system while it is in use will be introduced, and no

further safety analysis of the functionality is required. The RollSmart sensor package is not a safety-critical system, and it makes no claims of compliance or attempts at conformance to development standards regulating such systems, including IEC:61508, IEC:62304, and IEC:60601.

8 Conclusion

Creating the RollSmart is a project which requires the exploration of various engineering principles, such as 3D printing, biomedical sensors and their implementations in microprocessor applications, cloud data storage, data processing, and CI/CD. This project aims to add sensors to a rollator that provide an abundance of useful clinical data for Doctors to determine treatment efficiency and monitor user health over time. These additional electronic components will be hidden as best as possible on the rollator to minimize the disruptions to the user and have been chosen to minimize the cost and energy requirements. The data provided by the RollSmart will allow doctors to have a better understanding of their user's daily movements and exercise patterns and how their biometrics are affected by rollator usage. With its modular design, it can be retrofitted between different styles of rollators, which will in turn upcycle the rollator to sustain the environment. This project will offer the creators of RollSmart an adventurous experience and training in engineering projects. With that, group 33 is proud to present RollSmart.

8.1 Future Directions

Resource limitations and time constraints have resulted in the RollSmart project having a deliberately limited scope. We suggest the following viable "next steps" for future team members interested in improving on this design; these steps will help in improving the design and performance of the rollator, making it more efficient and effective for its target users.

1. **Conduct a Needs Assessment** Future researchers should conduct a needs assessment to identify the specific needs and requirements of the target users, providing more educated guidance on designing any new features for the rollator. The needs assessment should include conducting surveys, focus groups, and interviews with elderly users, healthcare professionals, and physical therapists to identify the most critical features required by the users. The data collected from the needs assessment should be analyzed to identify the most critical needs of the users, which will help in designing the new features for the rollator.

   The requirements for the current iteration of the RollSmart project were elicited without such feedback, which may limit the practical viability of some of the design decisions made.

2. **User Testing and Feedback Collection** User testing is crucial to ensure that the device meets the needs and expectations of the target users. Future researchers should conduct a series of user tests to gather feedback on the current design of the RollSmart rollator, its usability, and its functionality when used for practical purposes outside of a laboratory setting. The feedback collected can be used to improve the overall design of the rollator and ensure that the implementation decisions made for each elicited requirement meet the needs of the target users.

3. **Artificial Intelligence Integration** The data provided by the RollSmart sensor suite can be used to develop machine learning algorithms that can detect abnormal walking patterns, detect falls, and predict the likelihood of falls. The integration of artificial intelligence can improve the device's functionality and accuracy while providing users with real-time feedback on their gait and balance. AI-based fall detection systems may be especially relevant to improving the recovery prospects of individuals using rollators, as subtle differences in the forces applied to the rollator during a fall may be enough to allow the rollator to send a life-saving message for help in the event of an accident.

4. **Computer Vision and LIDAR Integration** The inclusion of LIDAR and computer vision sensors in the next iteration of the project by future researchers could significantly enhance the ability of the rollator to provide a safe and personalized walking experience for elderly users. LIDAR sensors could be used to create a 3D map of the environment, which would allow the rollator to detect and prevent collisions with obstacles and drops (i.e. stairwells) in real-time. The use of computer vision sensors would enable the rollator to recognize and track the user's movements, providing feedback on gait mechanics and balance.

5. **Voice-Activated Controls** Elderly users with disabilities may have difficulty using traditional controls. The integration of voice-activated controls can improve the device's accessibility and usability for these users.

References

[1] IEC SC 65A. Functional safety of electrical/electronic/programmable electronic safety-related systems. Technical Report IEC 61508, The International Electrotechnical Commission, 3, rue de Varembé, Case postale 131, CH-1211 Genève 20, Switzerland, 2018.

[2] Green Chan. Smart rollator prototype. *IEEE International*, page 1, 5 2008.

[3] Chan, Adrian DC, and James R. Green. Smart rollator prototype. IEEE International Workshop on Medical Measurements and Applications. IEEE, 2008., 2008.

[4] Vladimir Kulyukin, Aliasgar Kutiyanawala, Edmund LoPresti, Judith Matthews, and Richard Simpson. iwalker: Toward a rollator-mounted wayfinding system for the elderly. In *2008 IEEE International Conference on RFID*, pages 303–311, 2008.

[5] Dennis Lock. *Project Management (9th edition).* Ashgate Publishing Group, 2007.

[6] Nazanin Mansouri and Khaled Goher. Walking aids for older adults: Review of end-user needs. *Asian Social Science*, 12:109, 10 2016.

[7] Vítor Viegas Luisa Pedro Pedro Silva Girão Raul Oliveira Octavian Postolache, José Miguel Dias Pereira and Gabriela Postolache. Smart walker solutions for physical rehabilitation. *IEEE International*, page 4, 10 2015.

[8] Solenne Page, Ludovic Saint-Bauzel, Pierre Rumeau, and Viviane Pasqui. Smart walkers: an application-oriented review. *Robotica*, 35(6):1243–1262, 2017.

[9] Octavian Postolache, Jose Pereira, Vítor Viegas, Luisa Pedro, Pedro Silva Girão, Raul Oliveira, and Gabriela Postolache. Smart walker solutions for physical rehabilitation. *Instrumentation Measurement Magazine, IEEE*, 18:21–30, 10 2015.