



11/5/2021

Design Document

Balance Health



Name : Diarmuid Brennan
Student No : C00133947
Supervisor : Joseph Kehoe

Abstract

This document will describe the design manual for the Balance Health project. The project was designed to allow medical personnel to analyse and monitor a patient's performance while carrying out balance activities set by the medical personal which follow the 4-Stage-Balance test report.

The medical personnel will be able to monitor and analyse a patient's activities from the web application. The patient will retrieve the activities to their mobile device and carry out the activities while wearing the Movesense sensor device. The gathered sensor data will be sent to the mobile application and uploaded to the database for analysis.

This document will discuss the project architecture and technologies, the class and sequence diagrams used as well as providing wireframes to demonstrate the UI for the applications.

Table of Contents

Abstract	1
Table of Figures	4
Introduction	5
Architecture	6
Technologies	7
Movesense Sensor Device	7
Mobile Application	8
Web application	8
Firebase.....	9
Database Schema	10
Web Application.....	12
User Interface.....	12
Login.....	12
Register	13
Create Patient.....	14
View Patients.....	15
Create Activity	16
View Activities	17
View Activity Performance	18
Mobile Application	20
Class Diagram	20
Sequence Diagrams	21
Login.....	21
Logout	21
Register	22
Connect	22
Disconnect.....	23
Retrieve Activities.....	23
Perform Activity.....	24
Upload Results.....	24
View Progress	25
User Interface.....	26
Login Screen	26
Register Screen.....	27
Main Screen.....	28

Activity Details screen.....	29
Activity Description.....	30
Perform Activity screen.....	31
View Progress screen.....	32
Bibliography	33

Table of Figures

Figure 1	Project Architecture.....	6
Figure 2	Movesense Architecture	7
Figure 3	Login page	12
Figure 4	Register Page.....	13
Figure 5	Create Patient page	14
Figure 6	View Patients page	15
Figure 7	Create Activity page.....	16
Figure 8	View Activities page.....	17
Figure 10	View Activity Performance page	18
Figure 11	Login sequence diagram	21
Figure 12	Logout sequence diagram.....	21
Figure 13	Register sequence diagram.....	22
Figure 14	Connect to Movesense device sequence diagram	22
Figure 15	Disconnect from Movesense device sequence diagram	23
Figure 16	Retrieve patients activities from Firestore sequence diagram	23
Figure 17	Perform activity sequence diagram.....	24
Figure 18	Upload activity results to Firestore sequence diagram	24
Figure 19	View activity progress sequence diagram	25
Figure 20	Login Screen	26
Figure 21	Register screen	27
Figure 22	Main screen.....	28
Figure 24	Activity Details screen.....	29
Figure 25	Activity Details screen.....	30
Figure 26	Perform Activity screen	31
Figure 27	View Progress screen.....	32

Introduction

Pressures are continuously mounting on our healthcare system. A growing aging population will continue to add more stress to an already overwhelmed system. The Balance Health project was designed to create a solution that would allow medical staff to monitor a patient's balance performance remotely from a web application removing the need for many face-to-face interactions. The application will be used to aid in monitoring a patient's balance performance which can be used to determine early diagnosis into any deterioration in the patient's balance. This can then be used as a preventive measure in conditions such as risk of fall or other neurological issues for the patient.

The project consists of a web application, a mobile application, a cloud database and a Movesense sensor device. The medical personal can use the web application to track a patient's balance performance. The patient will then carry out the activities set from the Balance Health mobile application.

This document will discuss how the application is to be designed and used. It will describe the systems hardware and software architecture, contain class diagrams and sequence diagram to demonstrate the structure and interactions, as well as wireframes to illustrate the user interface for the applications.

Architecture

The technologies used in the development of the Balance Health project include a web application, a mobile application, a cloud database and a Movesense sensor device (**Figure 1**). Activities will be set by medical personnel for a patient on the web application and stored to the cloud database.

The patient will retrieve the activities from the database to their mobile device. While wearing the Movesense sensor device, the patient will perform the activities set and the application will retrieve the sensor data from the Movesense device. Once an activity has been completed the data is stored to the database from where the medical personal can retrieve the results and analyse the patient's performance.

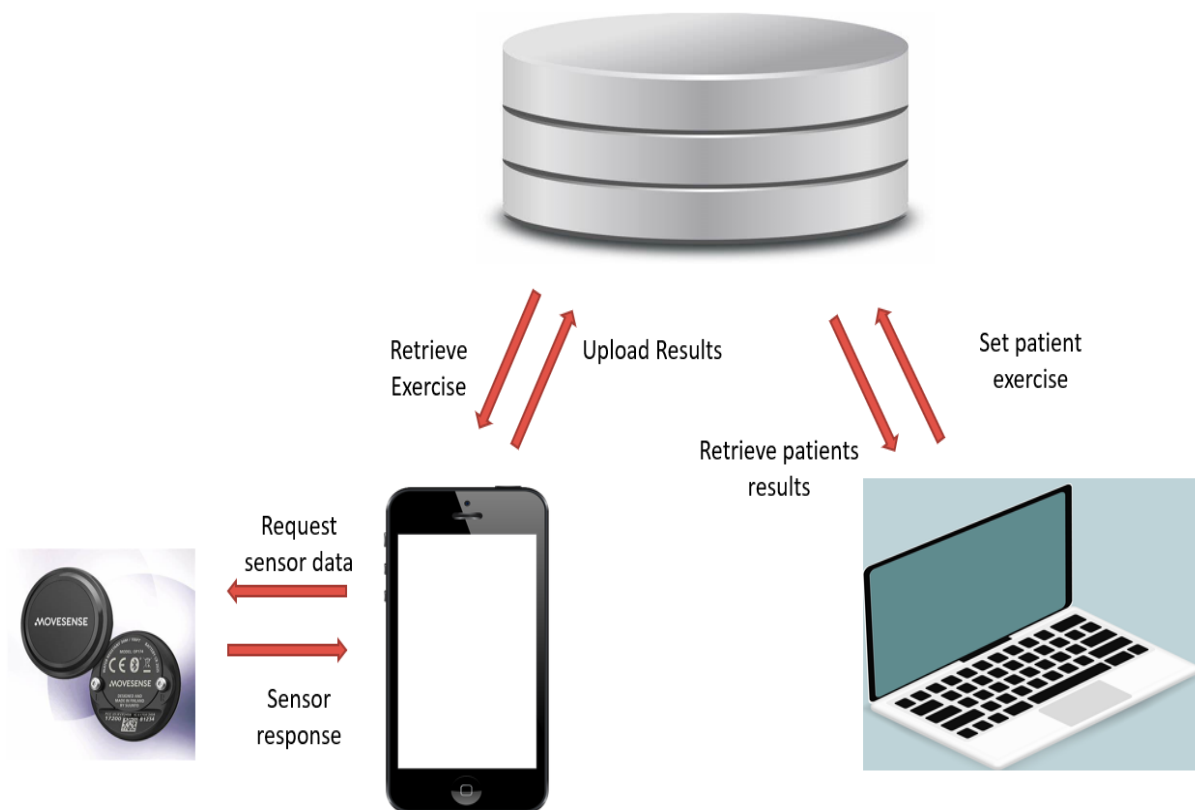


Figure 1 Project Architecture

Technologies

Movesense Sensor Device

The Movesense device is a small device consisting of several sensors. The sensors contained within the Movesense device include a 9-axis motion sensor, consisting of Accelerometer, Gyroscope and Magnetometer, a Maxim ECG Analog Frontend, which can measure ECG, heartrate and RR-intervals and a temperature measurement sensor. Data gathered through the sensors can be accessed over Bluetooth Low Energy (BLE) from an application installed on a mobile device. The Movesense device architecture (**Figure 2**) contains an inbuilt custom-made API service called Whiteboard and BLE services which are used to handle any requests and responses to and from the device.

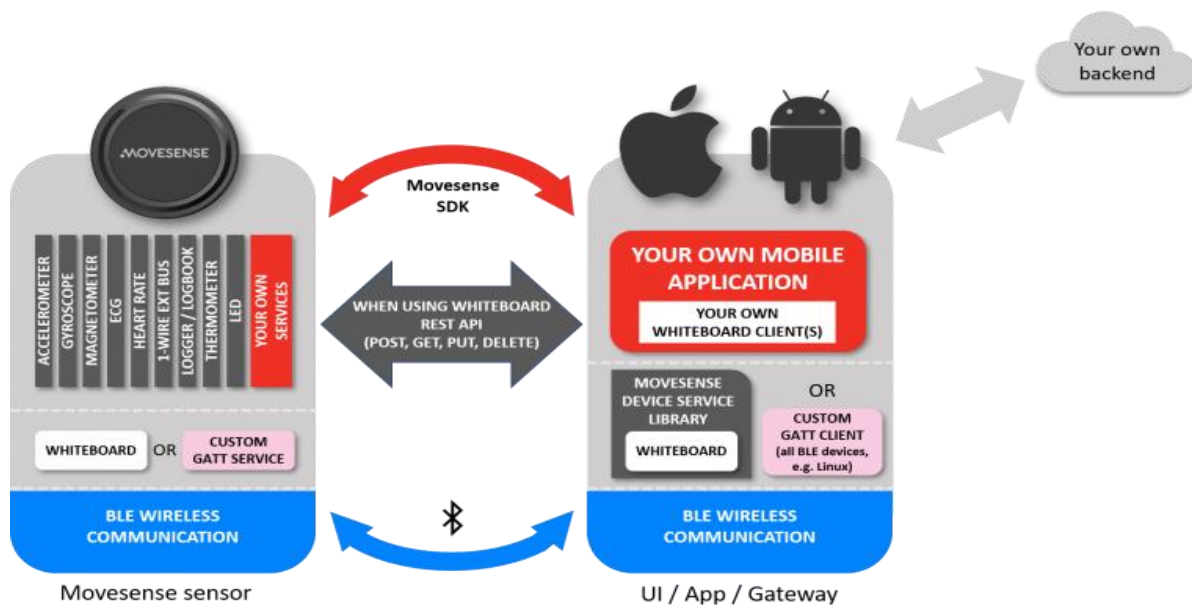


Figure 2 Movesense Architecture

The Whiteboard API provides the communication framework which allows for symmetric communication between the device and a connected mobile application. The Movesense platform provides the functionality for requesting data from the device allowing the developer the freedom to focus on developing the mobile application.

The patient will access the activities set by the medical personal from their mobile device. When an activity has been selected the patient will carry out the activity while wearing the Movesense device. Sensor data will be transmitted from the sensor device to the mobile application for the duration of the activity. On completion the mobile application will disconnect from requesting data from the device.

Mobile Application

The mobile application will be developed using the Java programming language for Android devices using Android Studio. Android studio is the official IDE for the Android operating system and is “the primary IDE for native Android application development” (Wikipedia, 2021). It provides the functions and services that will allow to create the mobile application.

To communicate with the sensor device, the Movesense platform provides the Movesense Device Service (MDS) library. The MDS library contains the components and interfaces needed to communicate with the sensor device over the Whiteboard communication framework. The MDS library allows the developer to create a Mds object which contains methods for connecting to and communicating with the sensor. Sensor data can be gathered from the sensor using the API request calls provided.

To connect to the device, the mobile application will also require a library that will provide the functionality to search for and connect with the device over BLE. This can be done using RxAndriodBle. RxAndriodBle is a powerful library that provides the functionality for scanning for nearby Bluetooth enabled and connecting to the device.

Once connected to the sensor device, the mobile application can subscribe to and request data from the sensor using the MDS library. The data can then be displayed on the device and stored to the cloud platform. Android Studio contains a Firebase Assistant which allows the developer to register their android application with a Firebase project. This adds the necessary Firebase files, plugins and dependencies required for the application to connect to the Firebase database.

A patient will create an account on the mobile application and the application retrieve activities set by the medical personal from the Firebase database. The activities will be displayed on the patients mobile, and the patient will carry out the activities while wearing the Movesense device. The data gathered from the device during the activity will be displayed on the device and the results will be stored to the database upon completion. The patient will also be able to view their progress for each activity on the application.

Web application

The web application will be developed using Python and the Flask web framework. Flask is a “lightweight Python web framework that provides useful tools and features that make creating web applications in Python easier” (DigitalOcean, 2021). Flask is a flexible and extensible framework containing an in-built server which allows developers to create web applications quickly and easily.

Flask in-built web server can handle incoming HTTP web requests and provide a response to the client. Flask uses the Jinja template engine to dynamically build HTML web pages using Python concepts. Jinja “is a Python template engine used to create HTML, XML or other markup formats that are returned to the user via an HTTP response” (Full Stack Python, 2021). Jinja templates contains variables that act as placeholders which can then store dynamic data rendered to the template from the python code. Jinja allows the developer to apply python programming logic in the templates.

A medical personal will be able to create an account on the web application. The web application will contain functionality for performing CRUD operations for a patient and set activities for the patients to carry out. The web application will be connected to the Firebase database using the Pyrebase library which allows for the web application to connect and communicate with the database.

Once activities have been set for a patient to carry out, the patient will download the activities to their mobile device, perform the activity and the results from the activity are stored to the database. The medical personal will be able to then retrieve the data from the database and monitor and analyse the patient's performance, setting new activities as the patient's performance improves or deteriorates.

Firestore

The cloud backend used for this project will be Firestore. Firestore is a Backend-as-a-service (BaaS) platform developed by Google. It provides the platform from which mobile and web application developers can link their applications to a backend cloud storage platform where data can be pulled directly from the cloud with no server involvement.

Firestore allows the developer to set up a project on the Firestore platform and register Apple, Android, or web applications with the project. Once an application has been registered, Firestore provides Firestore SDKs that allows access to Firestore products. Firestore products include, Cloud Firestore and Realtime database, Analytics, Performance monitoring, and Remote Config.

For this project it was decided to use the Cloud Firestore database. "Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firestore and Google Cloud" (Firestore, 2021). Firestore is a NoSQL cloud database which can be accessed directly from a mobile or web application using native SDKs. Firestore provides a flexible, scalable, and efficient method for storing and accessing the data including in-built security using Firestore Authentication.

Firestore allows you to store data in documents that contain key value mappings. Documents can store many different datatypes including strings, numbers, Booleans, maps, arrays, or timestamps. Documents are then stored in collections which act as containers for the documents. Documents can also store subcollections within to store further data and each of the collections, documents, and subcollections can be then queried individually for more efficient and flexible requests.

Database Schema

The database schema will contain the collections and documents needed to store, access, and manipulate the data required for the both the web and mobile applications. The schema for the applications is demonstrated below.

patient

- User_ID
 - o firstname
 - o lastname
 - o email
 - o userID

- A patient will be able to create an account on the mobile application.

medical_staff

- User_ID
 - o firstname
 - o lastname
 - o email
 - o userID

- Medical personnel will be able to create an account on the web application.

patients

- medical_staff_ID
 - o patient_ID
 - firstname
 - lastname
 - age
 - email
 - condition

- Medical personnel will be able to create a patient and add their medical information.

activities

- activity_ID
 - name
 - description
 - time_limit

- Medical personnel will be able to set activities for a patient to carry out. The patient will be able to retrieve the activities from the database to their mobile application.

patient_scores

- patient_ID
 - score_id
 - activity
 - [accelerometerData]
 - Avg_value
 - Max_value
 - Min_value
 - Date_set
 - activityNmae
 - completed

- A patient will be able to carry out activities and store their balance data results for each activity. The medical personal will be able to retrieve the results from the database to monitor and analyse the patient's progress.

Web Application

User Interface

Login

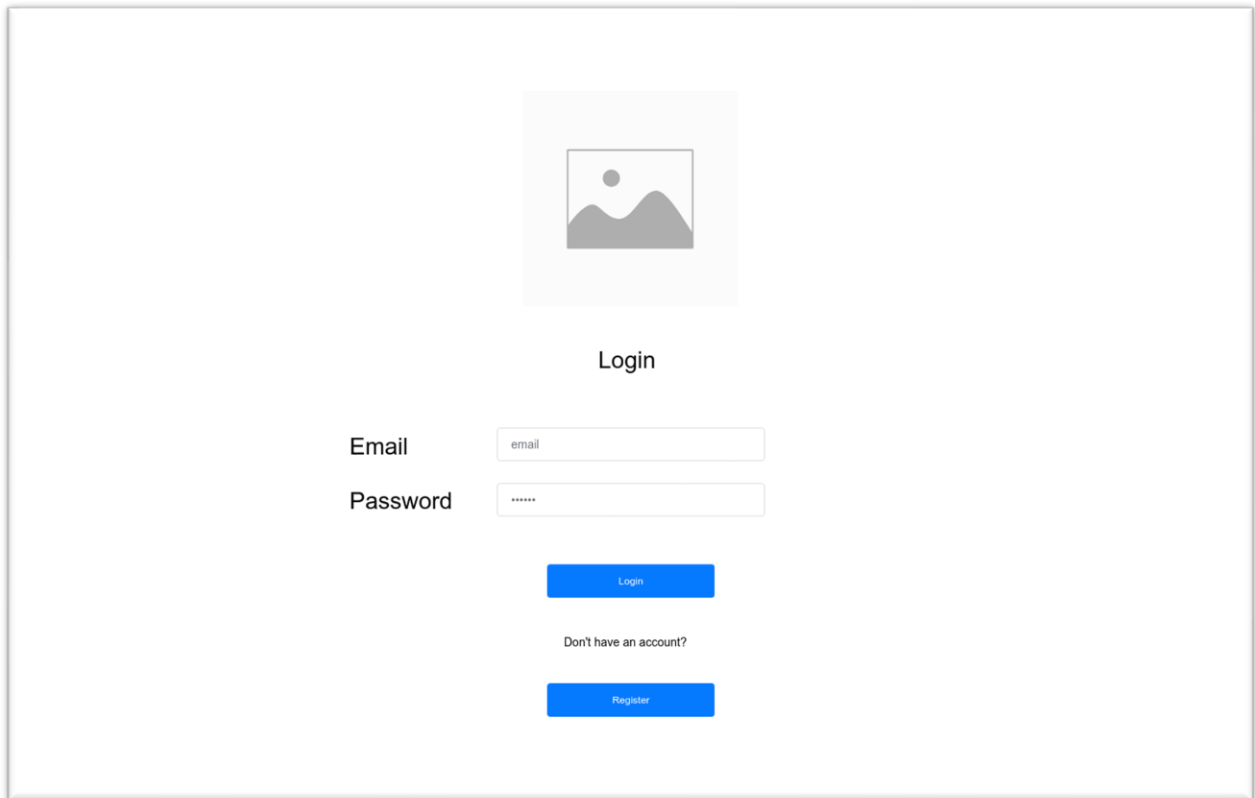


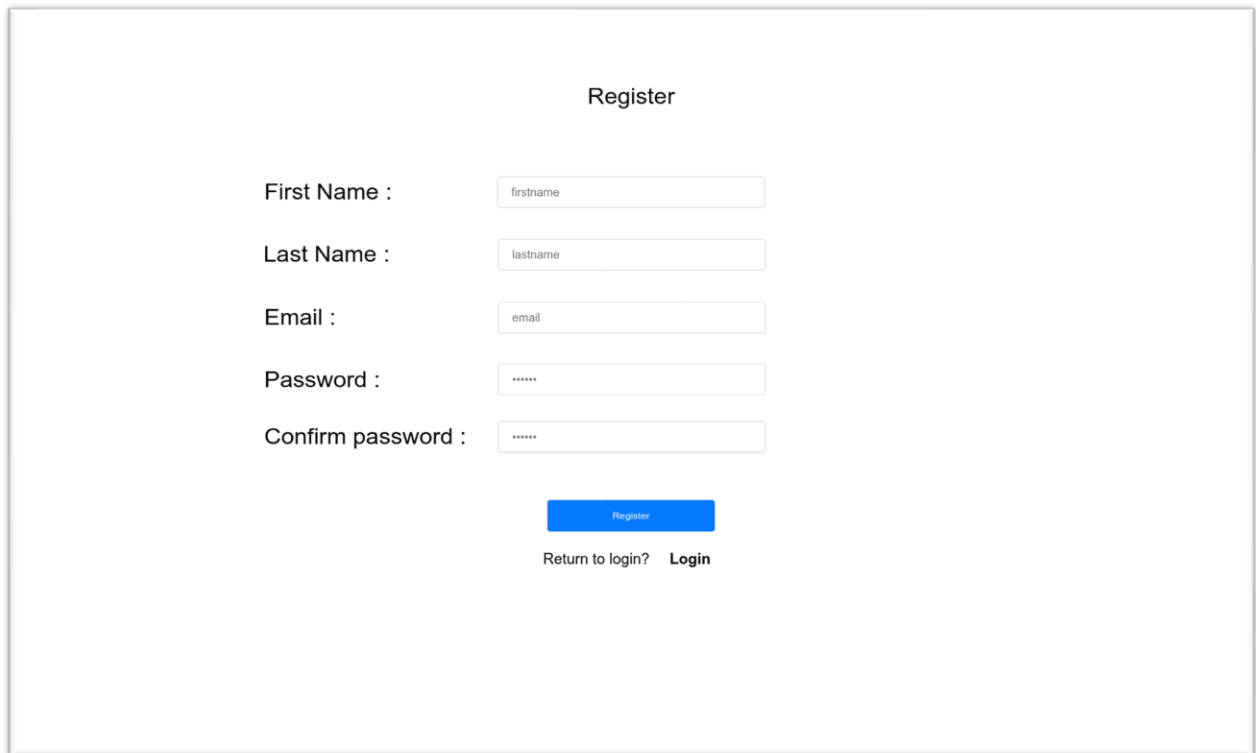
Figure 3 Login page

Medical personnel will be able to log on to the web application. The user's details will be stored using the Firebase Authentication procedure.

Login command

```
auth.sign_in_with_email_and_password(user_details["email"], user_details["password"]);
```

Register



The image shows a registration form titled "Register". It contains five input fields: "First Name" with placeholder "firstname", "Last Name" with placeholder "lastname", "Email" with placeholder "email", "Password" with placeholder "*****", and "Confirm password" with placeholder "*****". Below the fields is a blue "Register" button. At the bottom, there is a link "Return to login?" followed by a "Login" button.

Figure 4 Register Page

To create an account the user can access the register account page. Here the user can enter their details and Firebase authentication procedure creates a log in account for the user to access the website.

Register command

```
auth.create_user_with_email_and_password(user_details["email"], user_details["password"])
```

Create Patient

The screenshot shows a web application interface for creating a patient. At the top, there is a navigation bar with 'Balance Health' on the left, 'Patient' and 'Activity' dropdown menus in the center, and a 'Logout' button on the right. The main content area is titled 'Create Patient' and contains five input fields, each with a label and a placeholder text: 'First Name' (placeholder: 'firstname'), 'Last Name' (placeholder: 'lastname'), 'Email' (placeholder: 'email'), 'Age' (placeholder: 'age'), and 'Condition' (placeholder: 'condition'). Below these fields is a blue button labeled 'Create'.

Figure 5 Create Patient page

Medical personnel will be able to create patients, entering the patient's detail which are then stored to the database for that patient.

Add patient command

```
db.collection(u'patients').document(userid).collection(u'patient_details').document(user_details['email']).set({  
    u'firstname': user_details['first_name'],  
    u'lastname': user_details['last_name'],  
    u'email': user_details['email'],  
    u'age': user_details['age'],  
    u'condition': user_details['condition'],  
    u'activities': []  
})
```

View Patients

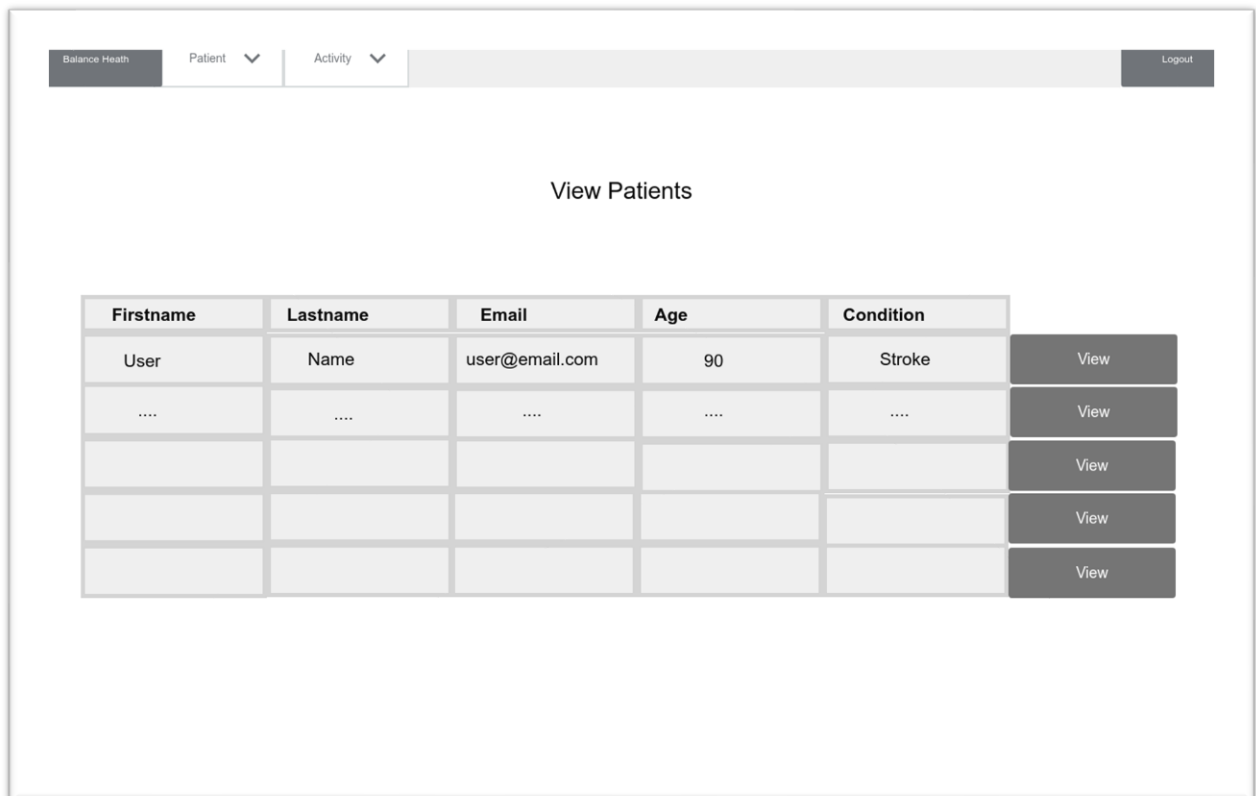


Figure 6 View Patients page

Medical personnel will be able to view each of their patients displayed in a table. Each patient row will contain a button that will bring the user to a page displaying the selected patients' personal details and activities.

Get patients command

```
docs = db.collection(u'patients').document(userid).collection(u'patient_details').stream()
patients = []
for doc in docs:
    patient = doc.to_dict()
    patients.append(patient)
return patients
```


Create Activity

The screenshot shows a web interface for creating an activity. At the top, there is a navigation bar with 'Balance Health', a 'Patient' dropdown menu, an 'Activity' dropdown menu, and a 'Logout' button. The main content area is titled 'Create Activity'. It contains three input fields: 'Name' (a text input field with the placeholder text 'name'), 'Description' (a text area with the placeholder text 'activity description'), and 'Time (seconds)' (a text input field with the placeholder text '30'). Below these fields is a blue button labeled 'Create'.

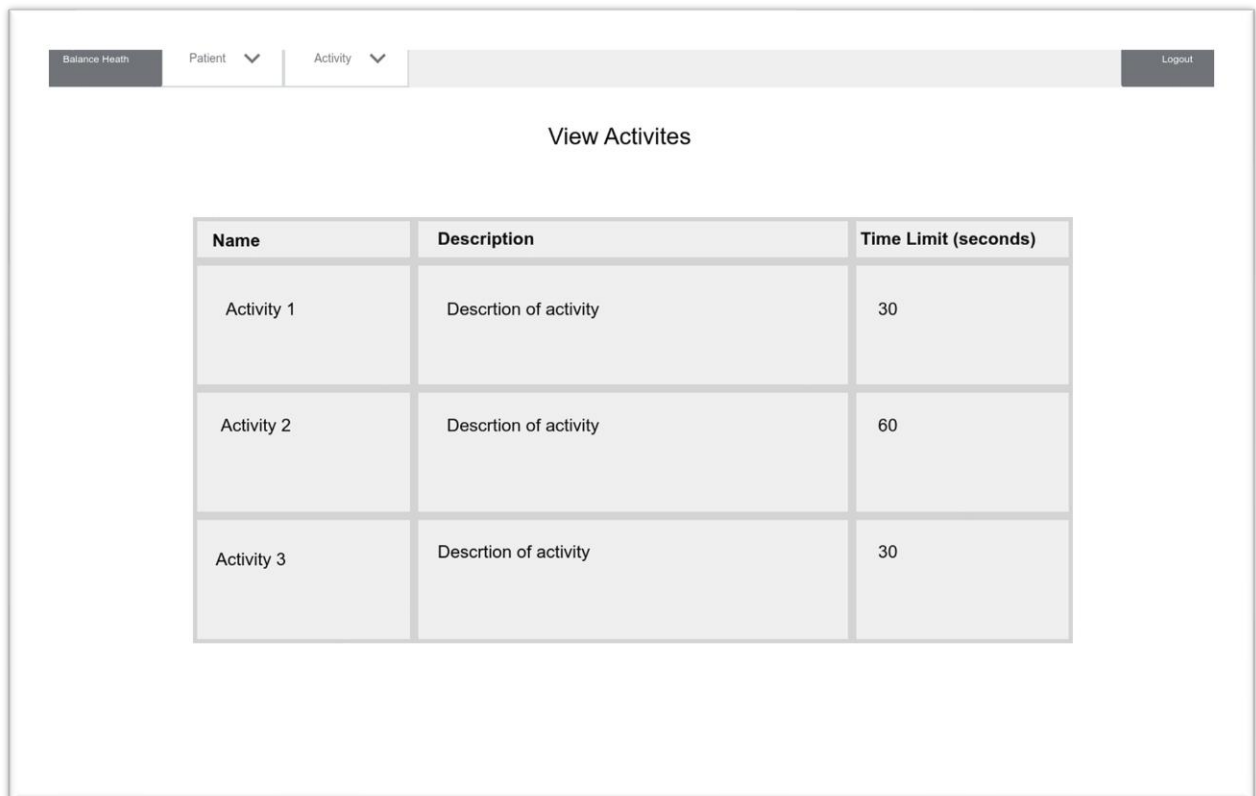
Figure 7 Create Activity page

Medical personnel will be able to create activities, adding the activities detail which are then stored to the database for that activity.

Add activity command

```
doc_ref = db.collection(u'activities').add({  
    u'name': activity_details['activity_name'],  
    u'description': activity_details['description'],  
    u'time_limit': activity_details['time_limit']  
})
```

View Activities



Name	Description	Time Limit (seconds)
Activity 1	Descrtion of activity	30
Activity 2	Descrtion of activity	60
Activity 3	Descrtion of activity	30

Figure 8 View Activities page

Medical personnel will be able to view each of their activities displayed in a table. Each row will contain the details for each activity.

Get activities command

```
docs = db.collection(u'activities').stream()
activities = []
for doc in docs:
    activity = doc.to_dict()
    activities.append(activity)
return activities
```

View Activity Performance

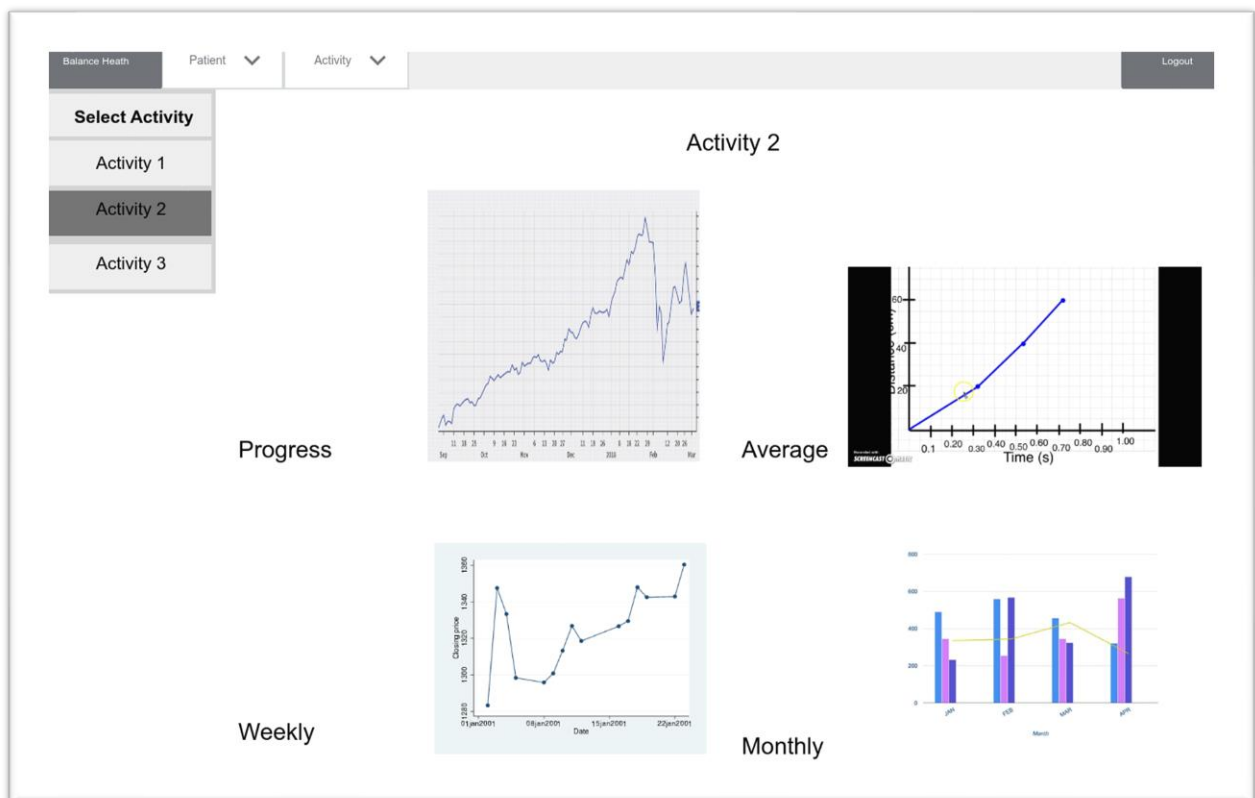


Figure 9 View Activity Performance page

Medical personnel will then be able to select activity from the patient's set activities and view the patients progress when carrying out that activity.

Get patients activities command

```
patient_activities = []
```

```
docs = db.collection(u'patient_activities').document(email).collection(u'activities').stream()
```

```
for d in docs:
```

```
    activity = d.to_dict()
```

```
    patient_activities.append(activity)
```

```
return patient_activities
```

Get activity results command

```
docs =
db.collection(u'patient_activities').document(email).collection(u'activities').document(activity).collection(u'scores').stream()

    scores = []

    for doc in docs:

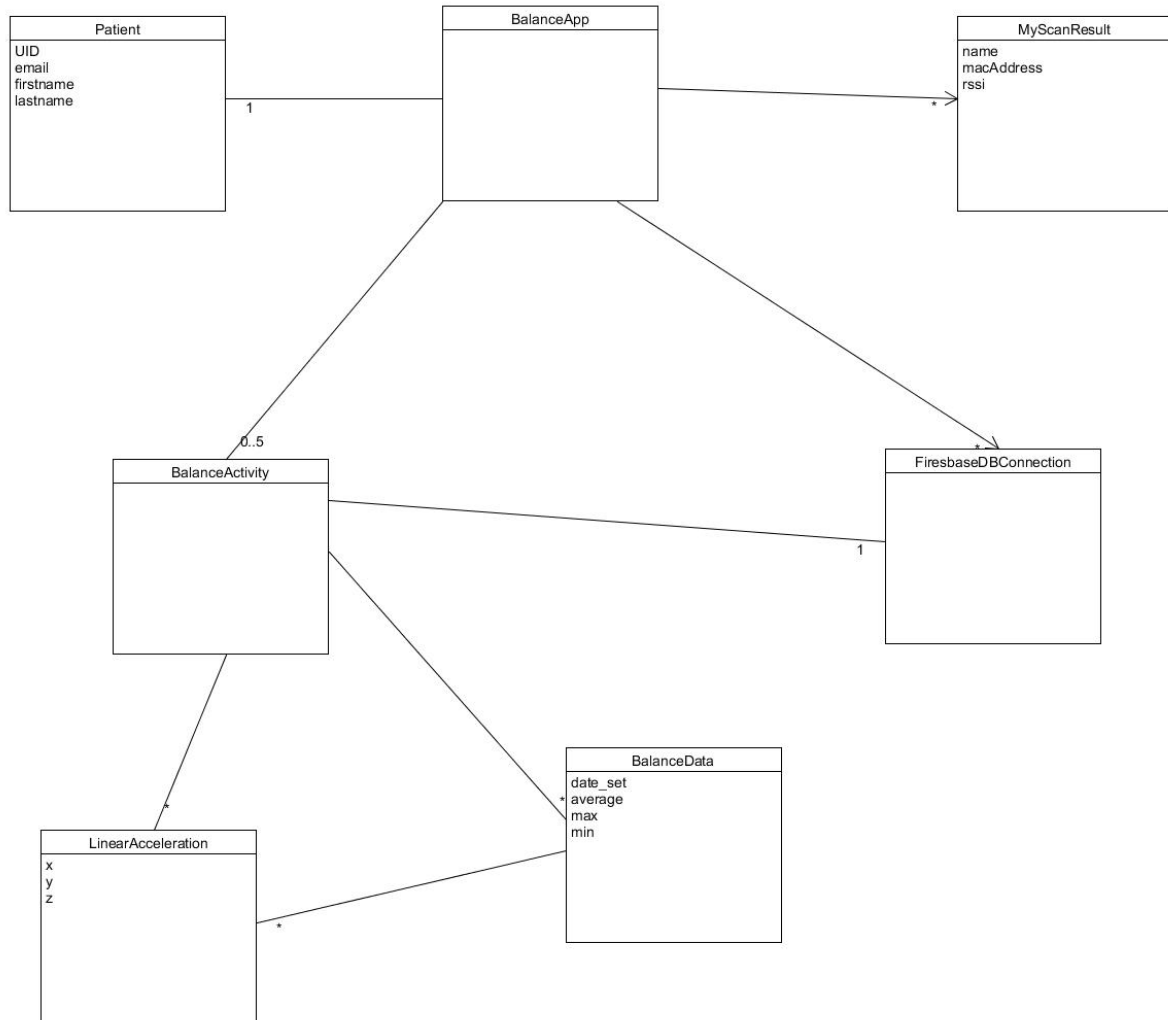
        activity = doc.to_dict()

        scores.append(activity)

return scores
```

Mobile Application

Class Diagram



Sequence Diagrams

The sequence diagrams will demonstrate the main use cases outlined in the Functional Specification document.

Login

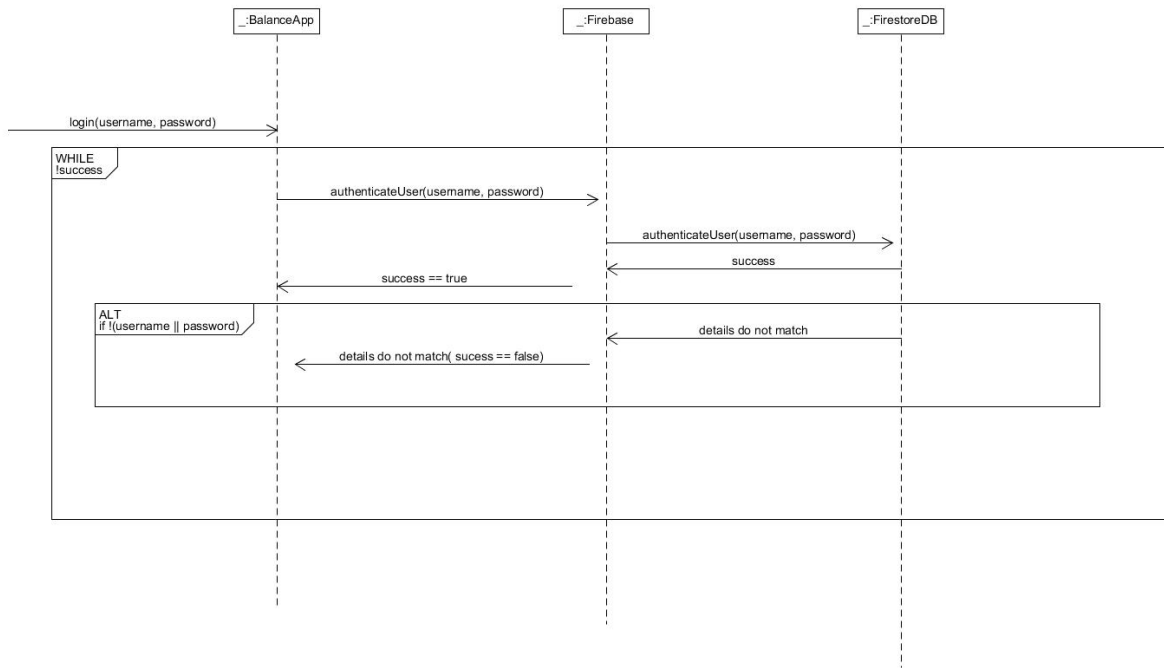


Figure 10 Login sequence diagram

Logout

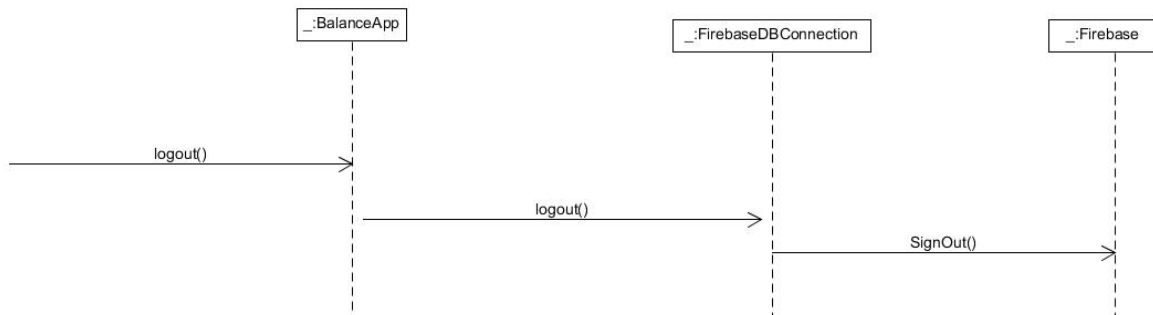


Figure 11 Logout sequence diagram

Register

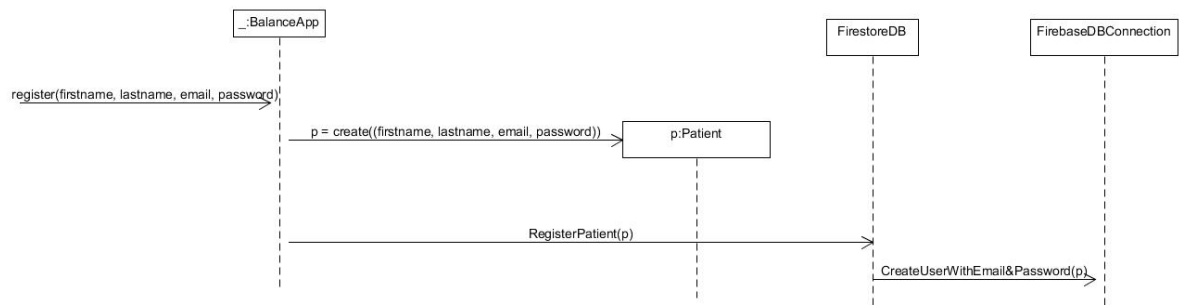


Figure 12 Register sequence diagram

Connect

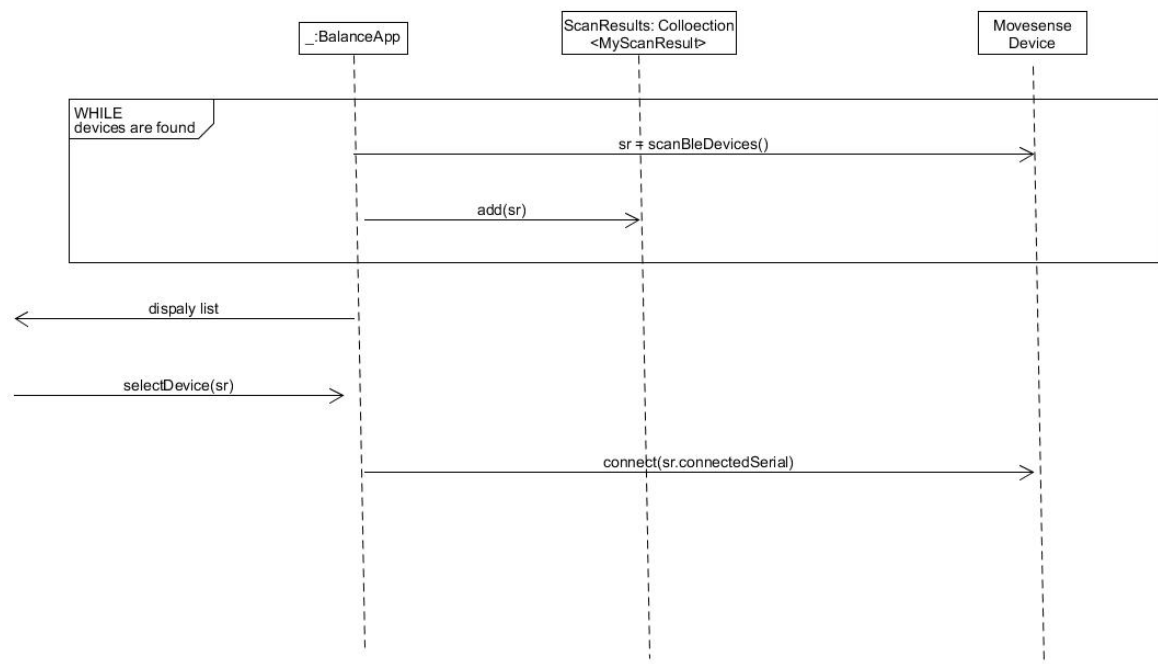


Figure 13 Connect to Movesense device sequence diagram

Disconnect

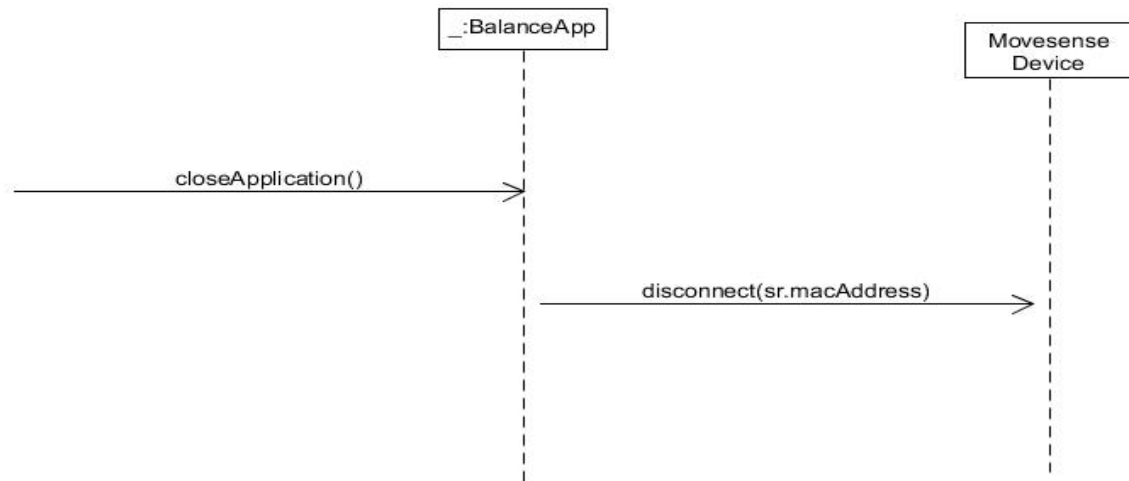


Figure 14 Disconnect from Movesense device sequence diagram

Retrieve Activities

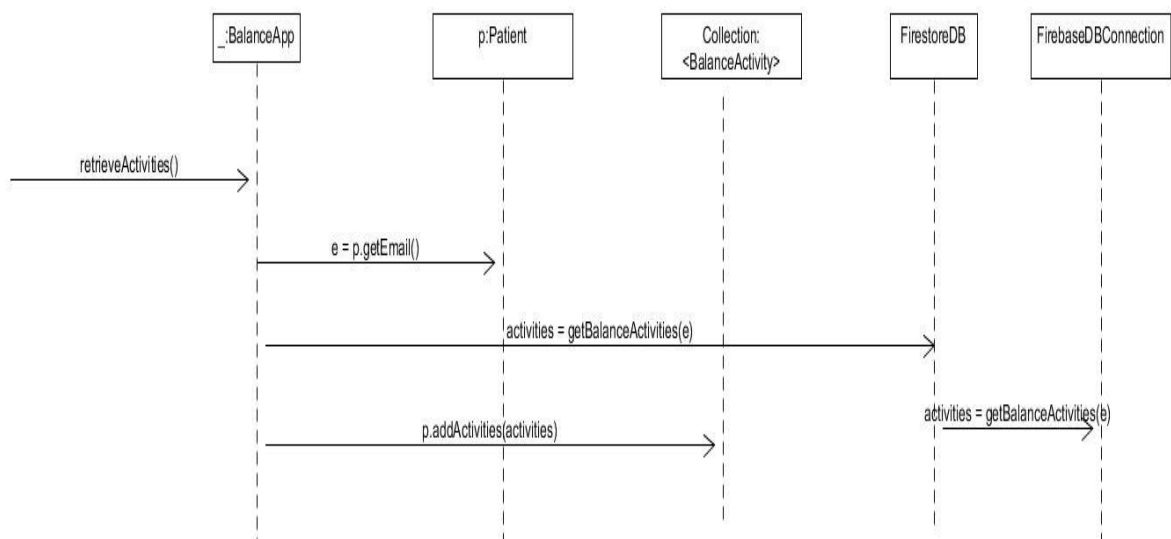


Figure 15 Retrieve patients activities from Firestore sequence diagram

Perform Activity

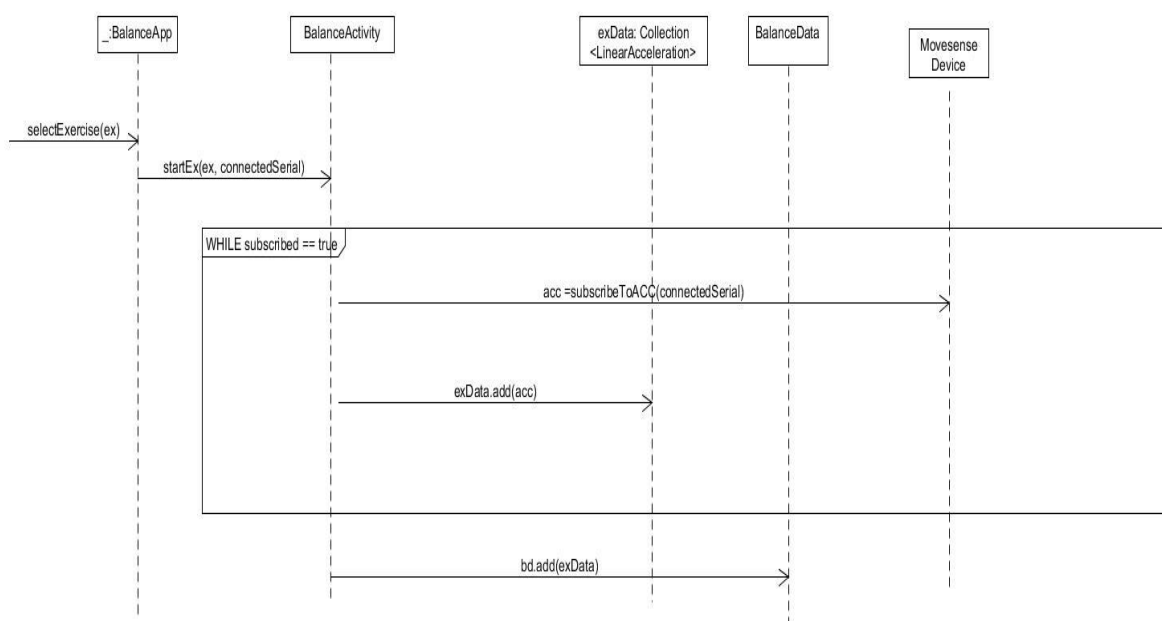


Figure 16 Perform activity sequence diagram

Upload Results

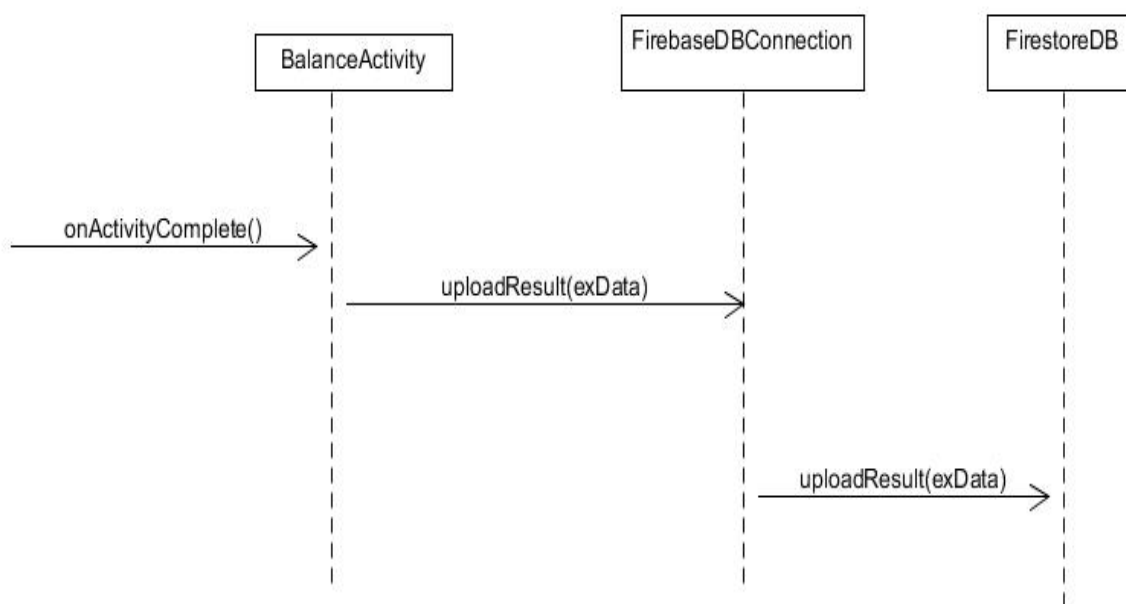


Figure 17 Upload activity results to Firestore sequence diagram

View Progress

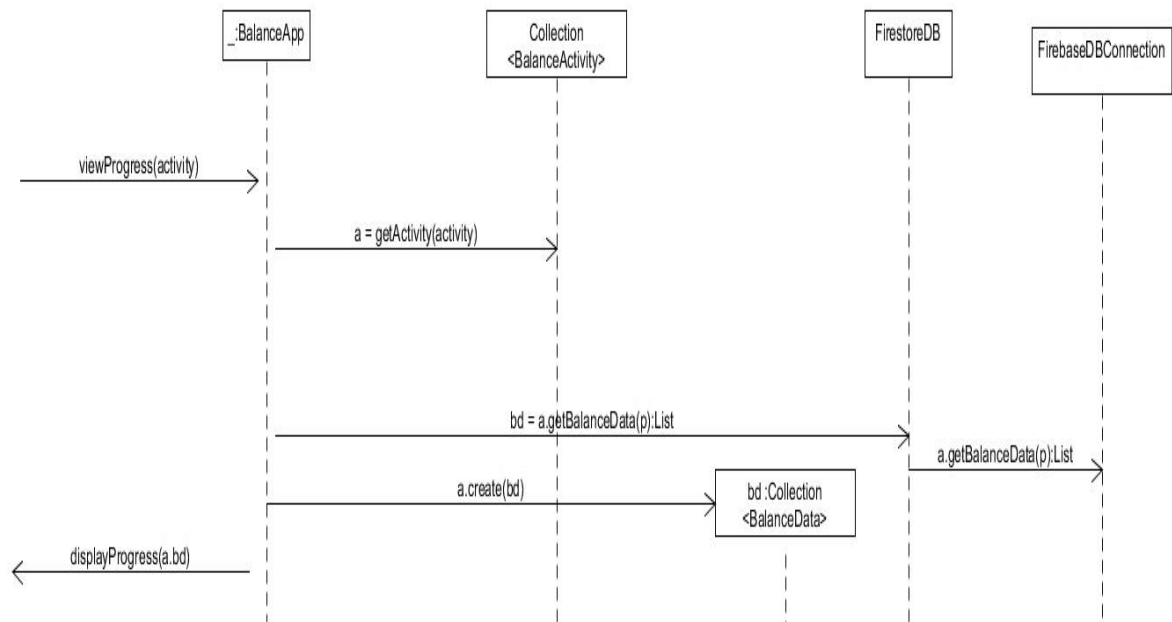


Figure 18 View activity progress sequence diagram

User Interface

Login Screen

The login screen will be the initial screen that the user will view upon opening the application. It will authenticate users using the entered username and password against the details stored on the database. If the user does not have an account, they can select the Register link and this will open the register screen.

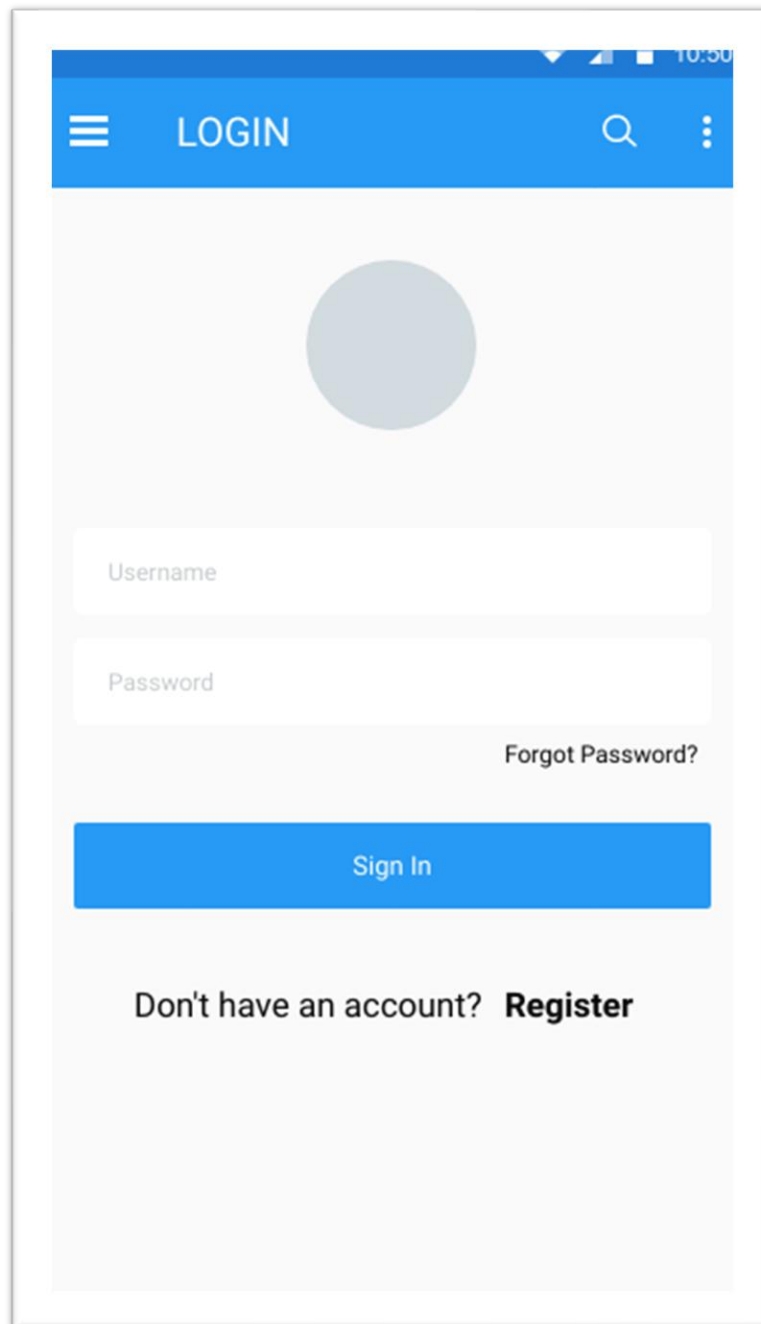
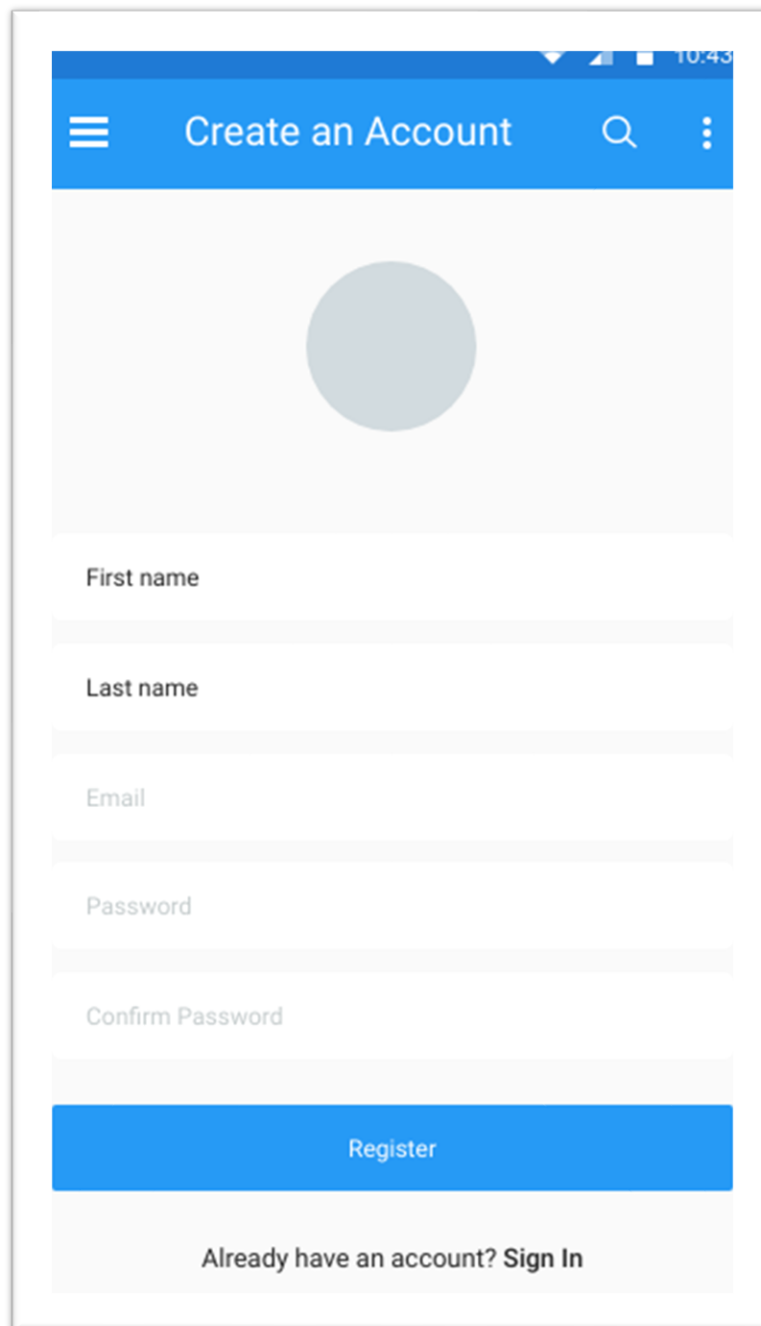


Figure 19 Login Screen

Register Screen

If a patient does not have an existing account, they can access the register screen from the login page. The register screen will allow the patient to create an account using Firebase's authentication procedure and will store the patient's details on the Firestore database.



The screenshot shows a mobile application interface for creating an account. At the top, there is a blue navigation bar with a hamburger menu icon on the left, the text "Create an Account" in the center, a search icon on the right, and a vertical ellipsis icon. Below the navigation bar is a large, light grey circle representing a profile picture. Underneath the profile picture are five input fields for registration: "First name", "Last name", "Email", "Password", and "Confirm Password". At the bottom of the form is a prominent blue button labeled "Register". Below the "Register" button, there is a link that says "Already have an account? Sign In".

Figure 20 Register screen

Main Screen

Once the patient has been successfully authenticated and is logged onto the application the patient will be sent to main screen. Here the application will carry out a scan to check for any nearby Bluetooth enabled devices. Any devices found containing Movesense within their path name will be displayed in a list on the page. The user can select the appropriate Movesense device, and the app will connect with the device. A message will display when the application has successfully connected to the device. The screen will also contain a button which will bring them to the Activity List screen.

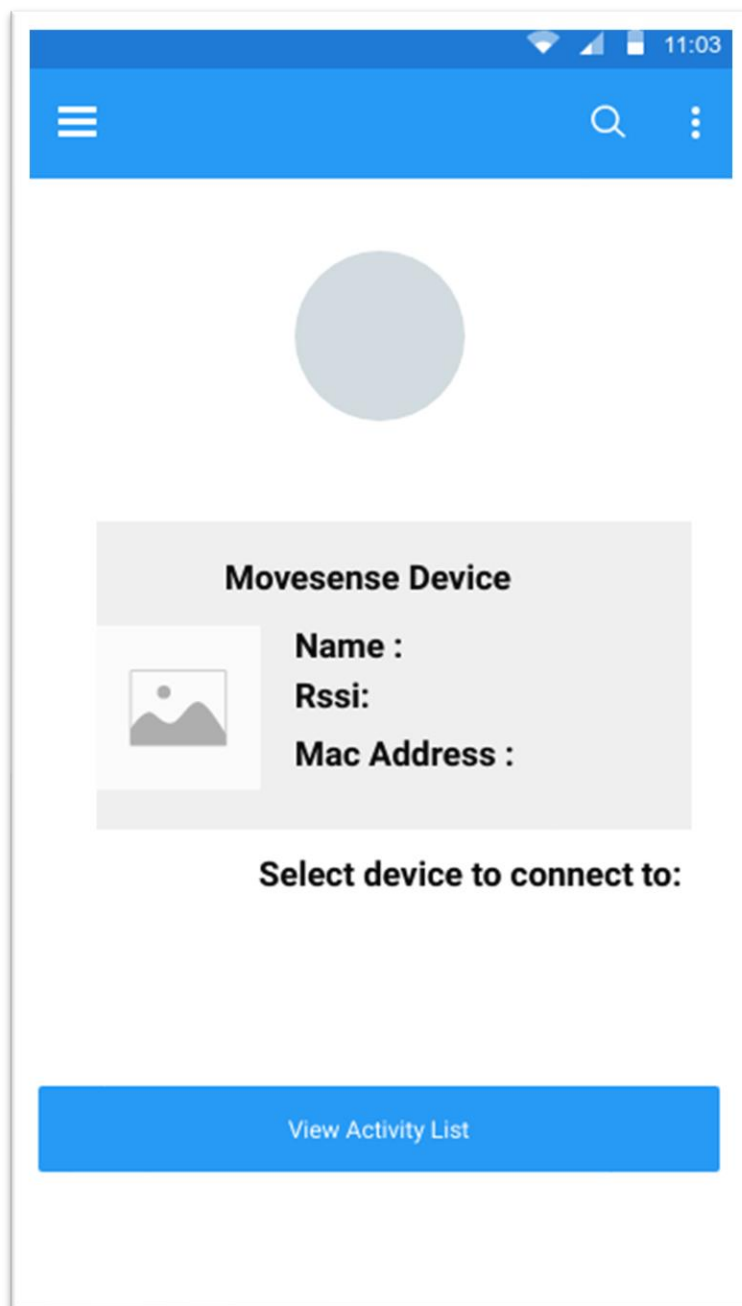


Figure 21 Main screen

Activity Details screen

Once an activity has been selected from the patient's activity list the patient will be brought to the activity details screen. The activity details will contain links that will allow the patient to view a description of the activity to be carried out, to perform the activity and to view their progress made when carrying out the activity.

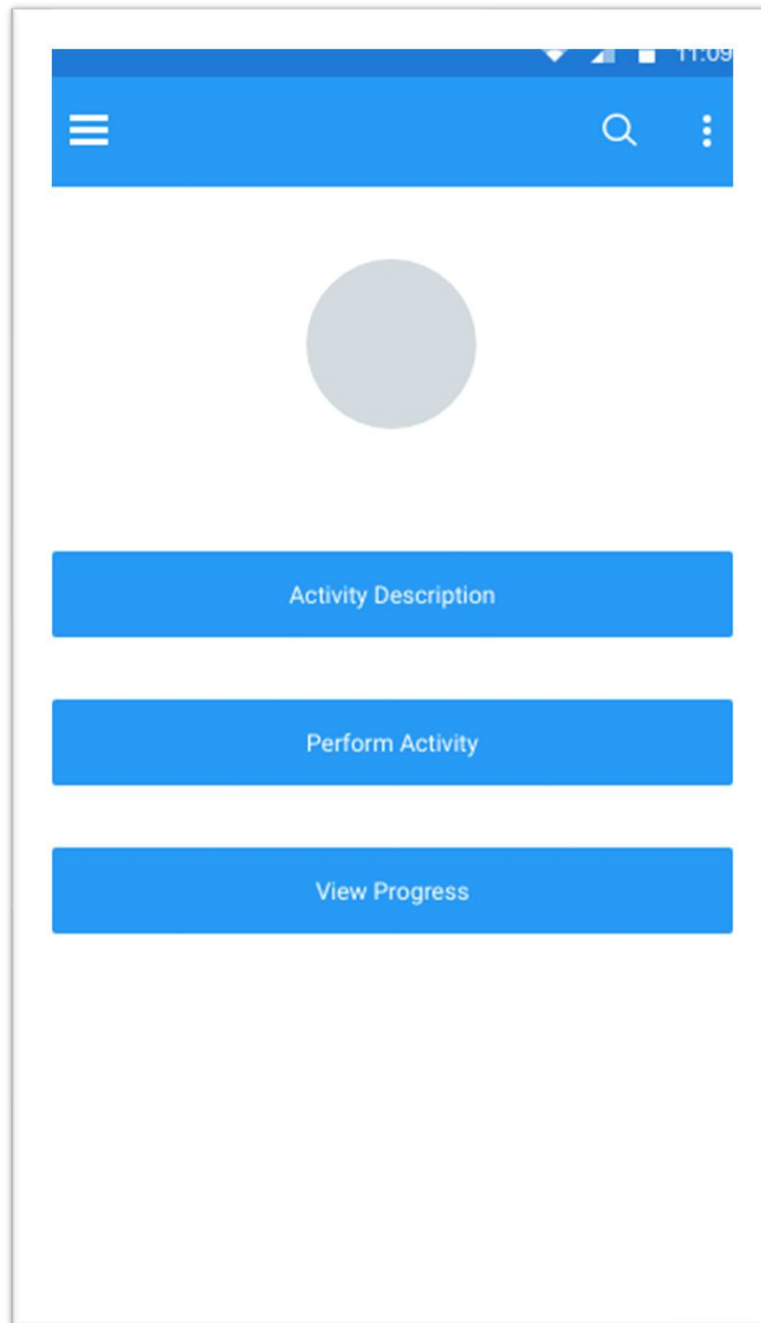


Figure 22 Activity Details screen

Activity Description

Upon selecting the activity description button the patient will be sent to the Activity Description screen. Here the description of the activity to be carried out will be detailed to the patient. The description will be retrieved from the database with the activities.

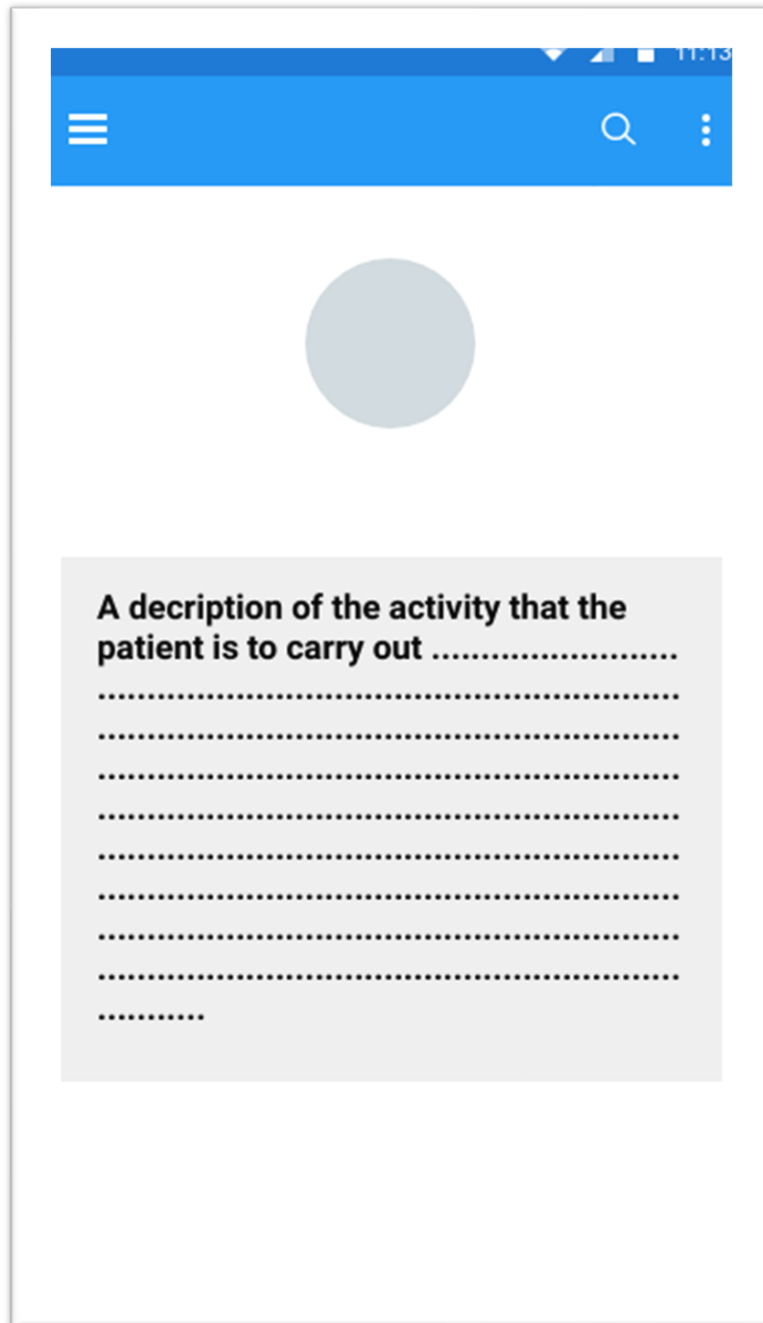


Figure 23 Activity Details screen

Perform Activity screen

On selecting the perform activity button the patient will be passed to the Perform Activity screen. Here the patient will carry out the activity described in the Activity Description screen while wearing the Movesense sensor device. The application will output a series of beeps alerting the user of when to begin the activity and another when to finish. Once the patient has been alerted to begin the activity, the application will subscribe to a sensor and begin requesting accelerometer data. The app will display the data to a chart and cancels the subscription once the activity has completed. Upon completion the gathered sensor data as well as some other additional information is uploaded to the Firestore database.

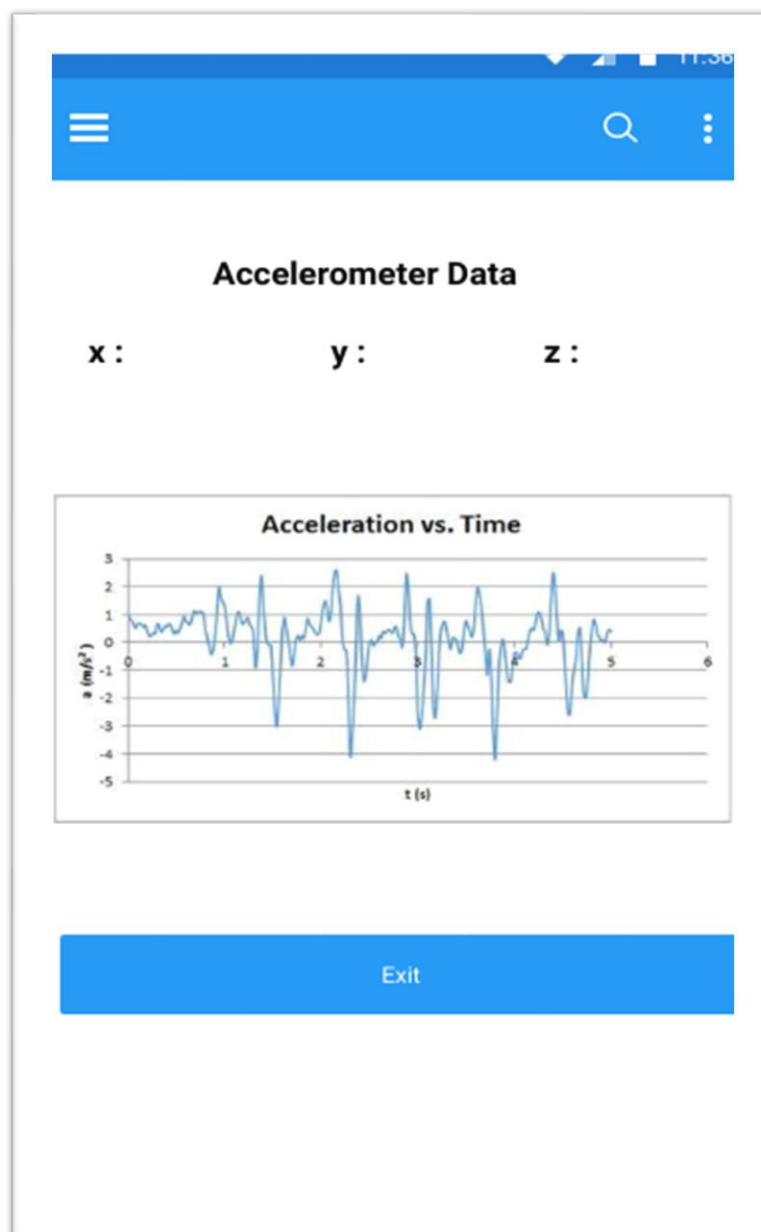


Figure 24 Perform Activity screen

View Progress screen

On selecting the View Progress button, the patient is brought to the View Progress screen. Upon opening the screen, the application will carry out a request to Firestore to gather the patient's previous results. An average score from each of patient's performances will have been calculated and the data will be displayed to a graph for the patient to view their progression while carrying out the selected activity.

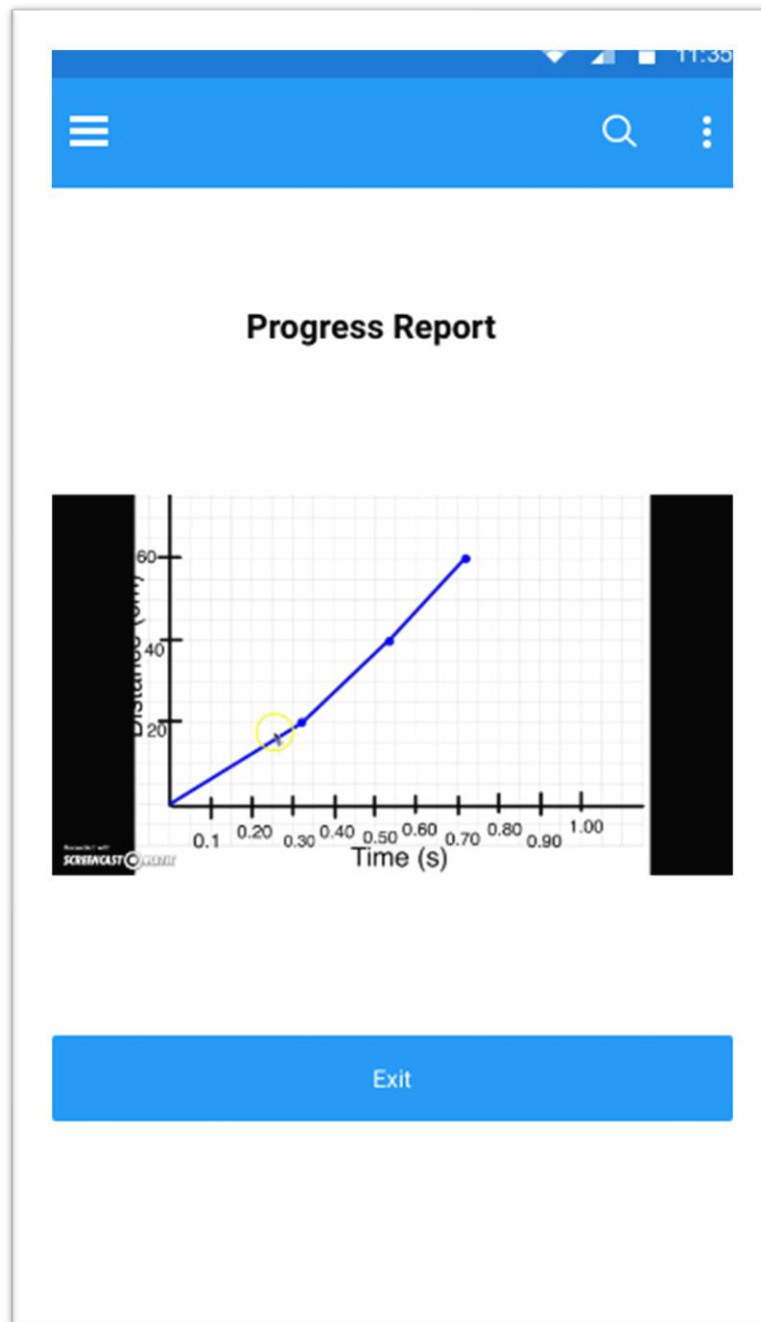


Figure 25 View Progress screen

Bibliography

DigitalOcean, 2021. How to make a Web Application using Flask in Python 3
Available at: [How To Make a Web Application Using Flask in Python 3 | DigitalOcean](#)
[Accessed 09 January 2022].

Firebase, 2021. Cloud Firestore
Available at: [Cloud Firestore | Firebase Documentation \(google.com\)](#)
[Accessed 09 January 2022].

Full Stack Python, 2021. Jinja2
Available at: [Jinja2 - Full Stack Python](#)
[Accessed 09 January 2022].

Wikipedia, 2021. Android Studio
Available at: [Android Studio - Wikipedia](#)
[Accessed 06 January 2022].

Wikipedia, 2021. How To Make a Web Application Using Flask in Python 3
Available at: <https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3> [Accessed 06 January 2022].