UNIVERSITY OF GHANA

DEPARTMENT OF BIOMEDICAL ENGINEERING

SCHOOL OF ENGINEERING SCIENCES

BMEN 400: FINAL PROJECT REPORT

MONITORING OXYGEN SATURATION(SpO_2) AND HEART RATE USING WEARABLE DEVICES

BY

DANIELLE PATRICIA QUAYE	10731732
RICHMOND TACKIE	10727463
BISMARK YAO AKWETEY	10709649
SAMUEL MPAABA WINBONO	10739147

SUPERVISOR: PROFESSOR ELSIE EFFAH KAUFMANN

CO-SUPERVISOR: DR. GODFREY MILLS

DATE: 12TH OCTOBER, 2022

Declaration of Originality

We declare that this thesis document is our own work except where indicated by references. We adhered to the guidelines and regulations spelled out by the University of Ghana policy on plagiarism. We acknowledged all sources used in this documentation and referenced all journals, articles, websites, and books by other people. To the best of our knowledge, the works of any student past or present of the University of Ghana and any other academic or research institution has not been replicated in our work.

STUDENTS:

DANIELLE PATRICIA QUAYE ……. ……..

Student ID:10731732 Signature Date

RICHMOND TACKIE …… ……..

Student ID:10727463 Signature Date

BISMARK YAO AKWETEY …… …….

Student ID:10709649 Signature Date

SAMUEL MPAABA WINBONO ……. ……..

Student ID:10739147 Signature Date

Certified by;

Professor Elsie Effah Kaufmann ....

(SUPERVISOR) Signature Date

Dr Godfrey Mills …… ……

(CO – SUPERVISOR) Signature Date

Acknowledgement

We appreciate the undeniable fact that this project would not have been possible without the support of a number of people. It is therefore an obligation to us to give honor to whom honor is due and appreciate all the inputs into this project.

First of all, we give thanks to the Almighty God who gives us life and in the goodness of His mercies, gave us strength, protection, sound minds and good health throughout our project.

We would also like to express our immense gratitude to our supervisor, Professor Elsie Effah Kaufmann for her exceptional supervision, advice, reprimands and corrections all in the attempt to shape us and encourage us to pursue our project passionately. We would like to thank our co-supervisor, Dr. Godfrey Mills of the computer engineering department for your consistency in revising our work and correcting us through every phase of our project. Your supervision was very prominent to the progress and success of our work.

We would also like to appreciate our collaborators, Dr. Hassan Ghomrawi and Dr. Matthew Glucksburg of the Northwestern University, Illinois, USA and Dr. William Appeadu- Mensah, head of the pediatric surgery unit at the Korlebu Teaching Hospital, Korlebu for their timely assistance and support throughout this project.

Our profound gratitude is expressed to our parents and guardians for investing greatly into our education and providing our basic needs so we could be comfortable and motivated to do this project.

We would like to acknowledge Ing. Lina A. Asante and the national service personnel of the clinical engineering department of the University of Ghana Medical Centre, UGMC for permitting the team to use their facility and equipment for the study.

We also acknowledge the supervision and encouragement given by Baron Afutu, a teaching assistant of the computer engineering department. This spurred us on during discouraging times of the project.

Finally, to all our course mates, friends and anyone who invested into this project through cash or kind, we are deeply grateful. God bless us all.

Abstract

Monitoring vital signs of patients help healthcare professionals to detect early signs of deterioration and aid doctors to make early-informed decisions on patients. In this pilot study, two smart wearable devices FITBIT VERSA 2 and XIAOMI MI BAND 6 were used to monitor vital signs of a healthy male volunteer of 22 years in a Ghanaian clinical setting. The data collected from the wearable devices were compared to a PHILIPS MX 450 patient monitor and a Balance brand BP device. The PHILIPS MX 450 patient monitor and the Balance brand BP device were used as the gold standards. The physiological data obtained were heart rate and SpO2 from the XIAOMI device and only heart rate from the FITBIT device.

Statistical analysis performed were the Bland-Altman analysis, Pearson correlation coefficient and the Mean Absolute Percentage Error (MAPE). Comparing the wearable devices to the PHILIPS MX 450 patient monitor, the Bland-Altman plot showed the following results; a mean difference of -0.86 bpm and limits of agreement (LoA) of (-10.56 to 8.82) for XIAOMI heart rate, a mean difference of 0.77 bpm and LoA of (-3.78 to 5.33) for FITBIT heart rate and a mean difference of -0.33% and LoA (-3.47 to 2.81) for XIAOMI SpO2 data. When compared against the manual monitoring device, the Bland-Altman plot showed the following results; a mean difference of -1.56 bpm and LoA of (-7.23 to 4.11) for FITBIT heart rate, a mean difference of -2.12 bpm and LoA of (-15.07 to 10.83). A Pearson correlation coefficient of 0.935949 and 0.749383 were recorded for FITBIT and XIAOMI respectively when compared against the PHILIPS MX 450 patient monitor. FITBIT and XIAOMI again recorded a Pearson correlation coefficient of 0.825244 and 0.422386 respectively when compared against the Balance brand BP device. The accuracy of the wearable devices were calculated using the MAPE of which FITBIT recorded an overall heart rate measurement accuracy of 96.87% against the PHILIPS MX 450 patient monitor and 96.64% against the Balance brand BP device. XIAOMI recorded a heart rate measurement accuracy of 94.24% when compared to the PHILIPS MX 450 patient monitor and 93.21% when compared to the manual monitoring device. XIAOMI again recorded an accuracy of 98.79% for SpO2 when compared to the PHILIPS MX 450 patient monitor. The team developed a web application to display the vital signs collected from the wearable devices in real time. The web application has the ability to display vital signs of at least 30 patients.

Table of Contents

List Of Figures

List of Tables

Chapter1 Introduction

1.1 Background and Literature Review

Wearable technology (WT) encompasses a myriad of devices either worn directly or loosely attached to a person [1]. WT can be subdivided into two categories, they include;

- Primary WT: These are WTs that operate independently and function as central connectors for other devices and/or information (eg. Wrist-worn fitness tracker)
- Secondary WT: These WTs operate/record specific actions or perform measurements. (eg. heart rate monitor worn around the chest off-loading to a primary wearable device for analysis.

Wearable Health Devices (WHDs) are emerging WT that enhance continuous monitoring of human physiological signs during daily activities or in a clinical environment with the benefit of lessening discomfort and/or interference with normal human activities [2]. There are many physiological signs that are measured in the human body; among these, there are five traditional vital signs that have a major significance to be measured: respiratory rate, heart rate, body temperature, blood pressure and blood oxygen saturation. Generally, it is considered imperative to measure these five vital signs since they are significant in human health evaluation. As such, there is a need for continuous measurement mainly in patients [3].

Vital signs are physiological data obtained from the human body, which describe the state of health of an individual. Continuous monitoring of vital signs enables changes in the trend of the vital signs to be noticed early and this can prevent complications that may lead to death [4], [5]. However, in many Ghanaian hospitals, nurses record vital signs in intervals of 4 hours, which results in the missing of early signs of deterioration in the vitals. In addition, night shifts of nurses can lead to loss of data, which is important for patient health evaluation [6]. As such, there is a need for a continuous ambulatory vital sign measurement to help prevent situations of deterioration that can easily be addressed if the trends of the vital signs were known.

Wearable Health Devices have been implemented in some hospitals as a supplement for vital sign measurement. In the USA, the ViSi Mobile device from Sotera Wireless [7] is a commercial product that has been accepted by the FDA as a health device and it is being used as a vital sign-monitoring device in some hospitals. It reads and records blood oxygen (SpO2)

levels, skin temperature, ECG, heart rate, blood pressure, and respiration/breathing rate [8]. In addition, SensiumVitals from Sensium Healthcare Limited [9] in the UK is an FDA-approved and CE-marked wearable vital sign monitoring system that records heart rate, breathing/respiration rate and axillary temperature [10].

In Ghana, the use of WHD has not been introduced yet into hospitals, patient monitors are the main devices used for vital sign monitoring of patients. Even with this, a lack in the supply of patient monitors for vital sign monitoring is prevalent. This then leads to manual monitoring of the vital signs. Manual monitoring of vital signs is faced with some challenges. These include; loss of data by nurses during shift change and lack of continuous ambulatory vital sign measurement during the night [6].

In view of this, this project seeks to use Fitbit Versa 2 and Xiaomi Mi band 6 to continuously measure the vitals of patients. Data recorded from these wearable technologies will then be compared to data from both patient monitors and manual modes of vital sign monitoring. The Fitbit Versa 2 measures temperature and heart rate while the Xiaomi Mi band 6 measures oxygen saturation (SpO2) and heart rate.

1.2 Problem Statement

There are challenges with the existing methods used in Ghanaian hospitals for monitoring patients' vital signs. Some of these challenges are as follows:

- High cost of patient monitors: In most hospitals, patient monitors are common for assessing vital signs of patients. The average price of one patient monitor is $4, 173 (₵45, 694.35). This is costly for low-income countries like Ghana. Therefore, many hospitals in Ghana are unable to purchase patient monitors as required for use.
- Inadequate supply of functioning patient monitors in Ghanaian hospitals. It is common to see very few functioning patient monitors in a Ghanaian hospital with some hospital wards sharing one or two functioning patient monitors. This is due to the inability to replace faulty monitors and the incapacity to fix them.
- High patient to nurse ratio makes manual monitoring difficult. The challenges encountered with using patient monitors has encouraged manual monitoring of vital signs. However, this puts a burden on the nurses who are unable to attend to the large number of patients. This approach is therefore very laborious and ineffective.

- Occurrence of error in manual measurement of vital signs. Monitoring vital signs manually comes with a lot of errors due to human error.

1.3 Need Statement

There is therefore the need for a more efficient and affordable method of monitoring the vital signs of patients.

1.4 Aim

This project aims at ascertaining the use of Fitbit Versa 2 and Xiaomi Mi Band 6 to monitor Oxygen Saturation (SpO2) and Heart rate in a Ghanaian hospital.

1.5 Objectives

1. To monitor the heart rate and oxygen saturation (SpO2) of at most 30 patients for the study.

2. To perform statistical analysis on the data and describe the concordance of the wearable technologies to the traditional mode of vital signs monitoring in the hospital.

3. To develop a platform (web application) that can display the data obtained from the wearable technologies.

1.6 Project Significance

Many preventable deaths result from rising complications because of the lack of continuous monitoring of vital signs of post-surgery patients. These complications are as a result of unnoticed changes in trends of vital signs by nurses on duty due to lack of continuous ambulatory vital signs measurement and inconsistency in vital signs measurement during night shift. This project seeks to help alleviate these problems by using Fitbit and Xiaomi WT for continuous vital signs measurement. The relevance of the project is outlined below;

1. The continuous measurement of vital signs will provide data on the trends of the vitals of post-surgery patients that will not be missed or unnoticed by nurses to enable the doctor to make informed and accurate diagnosis.

2. Design of a fully functioning web application that displays real time physiological data measured by the wearable devices to the clinicians

3. Contribution of information to literature on the methods of comparing the concordance of the wearable devices to the manual monitoring to help in advising the use of these devices in Ghanaian hospitals.

4. Manual palpitations by nurses to record vital signs, which is liable to human errors, will be discouraged.

1.7 Innovation

- Introducing the use of wearable technology in a Ghanaian hospital.
- A platform (web application) to allow health workers to obtain patient's physiological data wirelessly from the wearable devices in Ghanaian hospitals.
- Providing analysis on the performance of the wearable devices in relation to the patient monitor and manual monitoring of vital signs.

1.8 Approach

Web application development involves the use of client-side and server-side programming to design a program/application that is made accessible over the internet using a web browser. This innovative technology has been used to develop typical web applications for data collection and management. These applications store data through databases or files and manipulate data by allowing users to create, read, update and delete data, thus it allows input from users through the client-side and processes data through the server-side [11].

This project utilizes Python, Cascading Style Sheets (CSS), Hypertext Markup language (HTML), Structured Query Language (SQL), JavaScript (JS) and Bootstrap programming to develop a management system that can be used to visualize pulse oxygenation and heart rate obtained from Fitbit and Xiaomi wearables. The web application also incorporates other important functionality such as registering a new patient, uploading manual palpations, an alert

to show change in abnormal vital signs, a dashboard to view recent patient information and a report form for health personnel to input instructions. It also allows health personnel to create an account and verify their account before they are allowed to use the management system.

Chapter2 The Design Process

The management system should be able to retrieve SpO2 and heart rate data from Fitbit and Xiaomi wearables, thus, Application programming Interface tools (API) should be utilized. Therefore, the design process was divided into 4 steps to avoid repeating steps and mismatching requirements and hence, to develop a system that can be reproducible because of the quality of the code.

The steps are as follows:

2.1.1 Sketching

On a sheet of paper, the team sketched the web interface, indicating layout of forms, buttons, social media icons and brand logo placement. Discussion of how the navigation tab, search bar, and buttons would work and be visualized in conjunction with the welcome page, signup and sign in pages was carried out.

2.1.2 User Interface and User Experience (UI/UX)

In Figma desktop, the design team implemented the design concept. Figma is an interactive web platform for interface design, with desktop apps for macOS and Windows enabling additional offline functionalities. Color, transitions, graphics, photos, shapes, and effects were added, and a mock-up of the management system was created, which included some functionalities such as user sign-up, login sequence, visualizing patient vitals, navigating the application, and logging out.

The images below show the various pages on the web application conceptual design in Figma

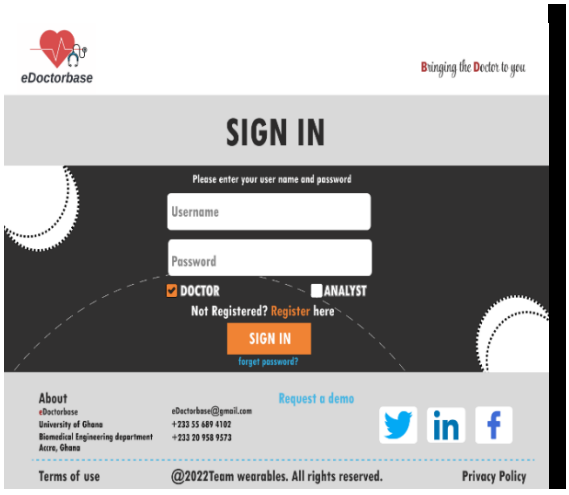


Figure 2.1 Sign in page for old users

Figure 2.2 Registration page for new users

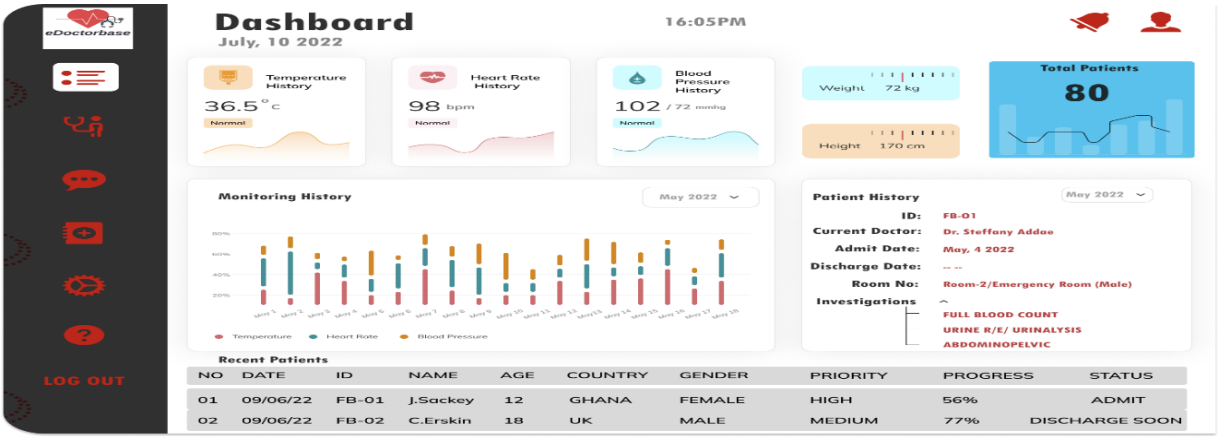


Figure 2.3 Dashboard for clinicians

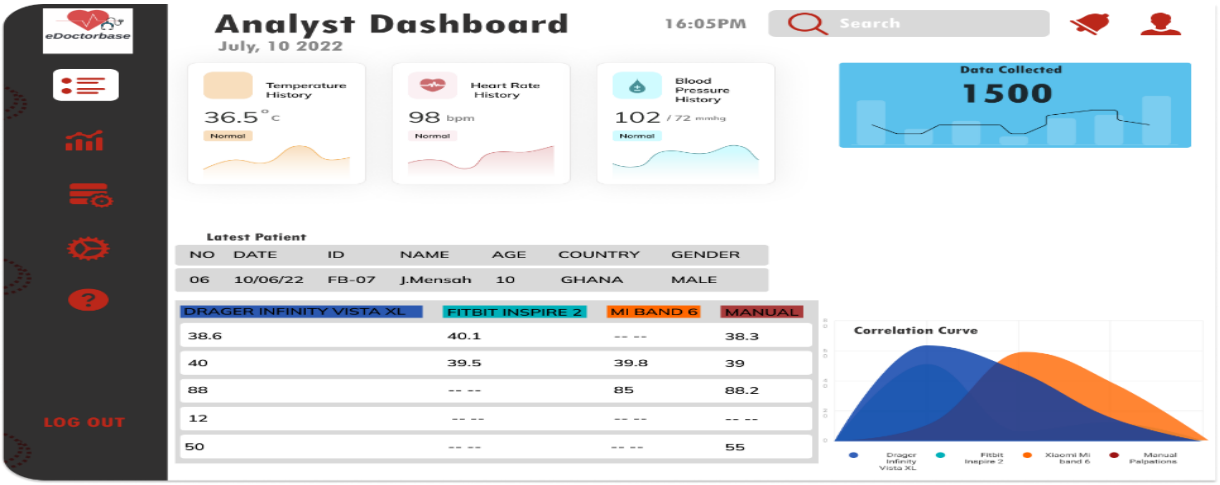


Figure 2.4 Dashboard for data analyst

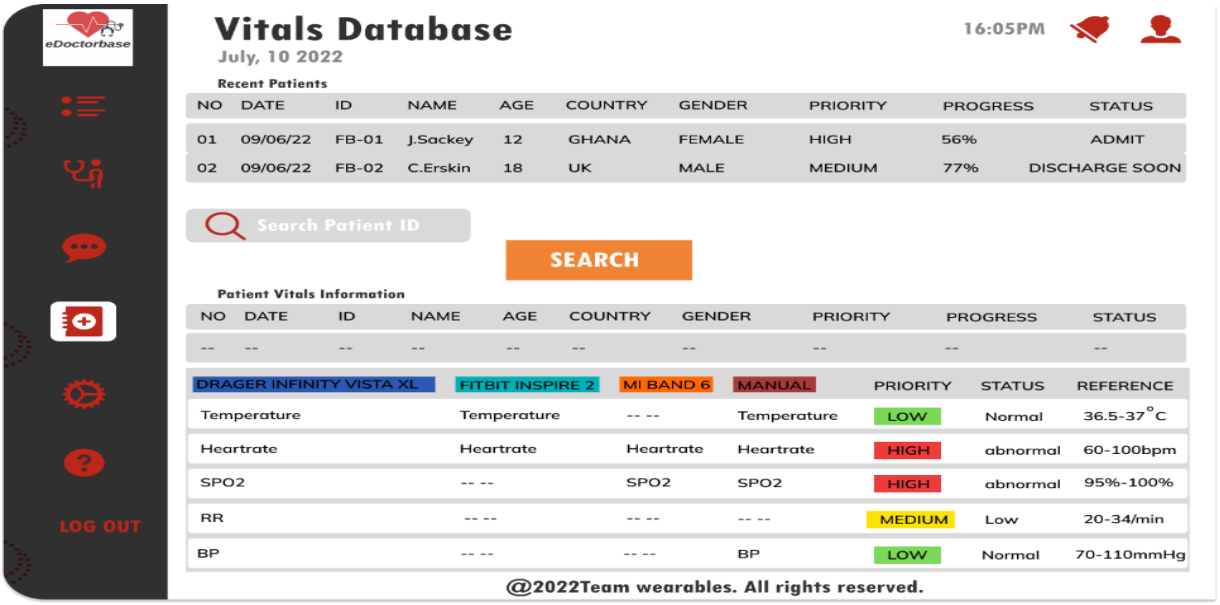


Figure 2.5 Database where all vitals recorded from patients are displayed

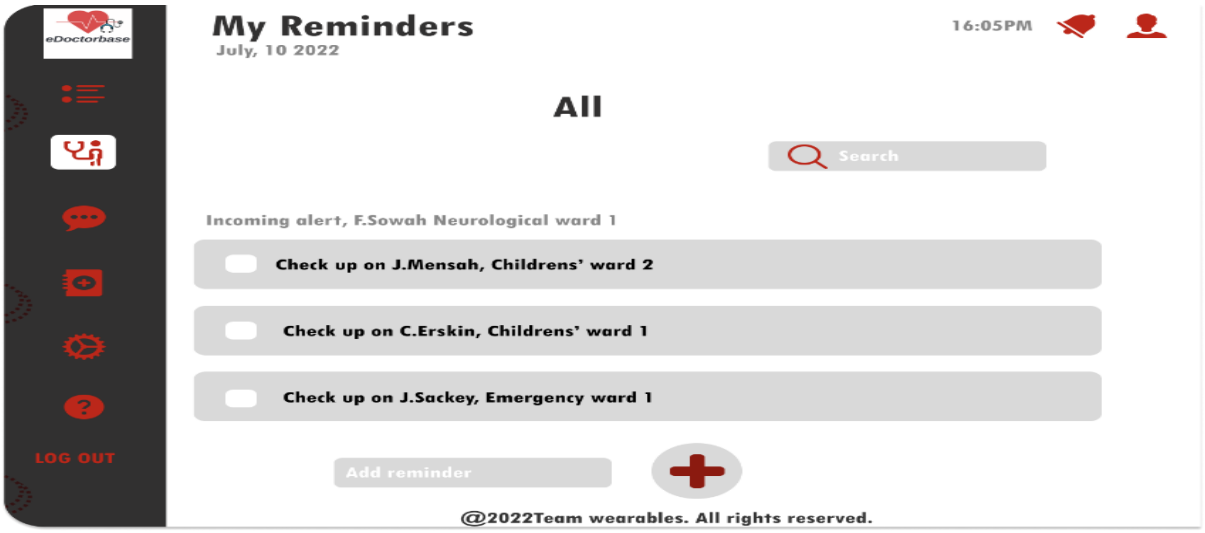


Figure 2.6 Reminders tab for setting alerts.

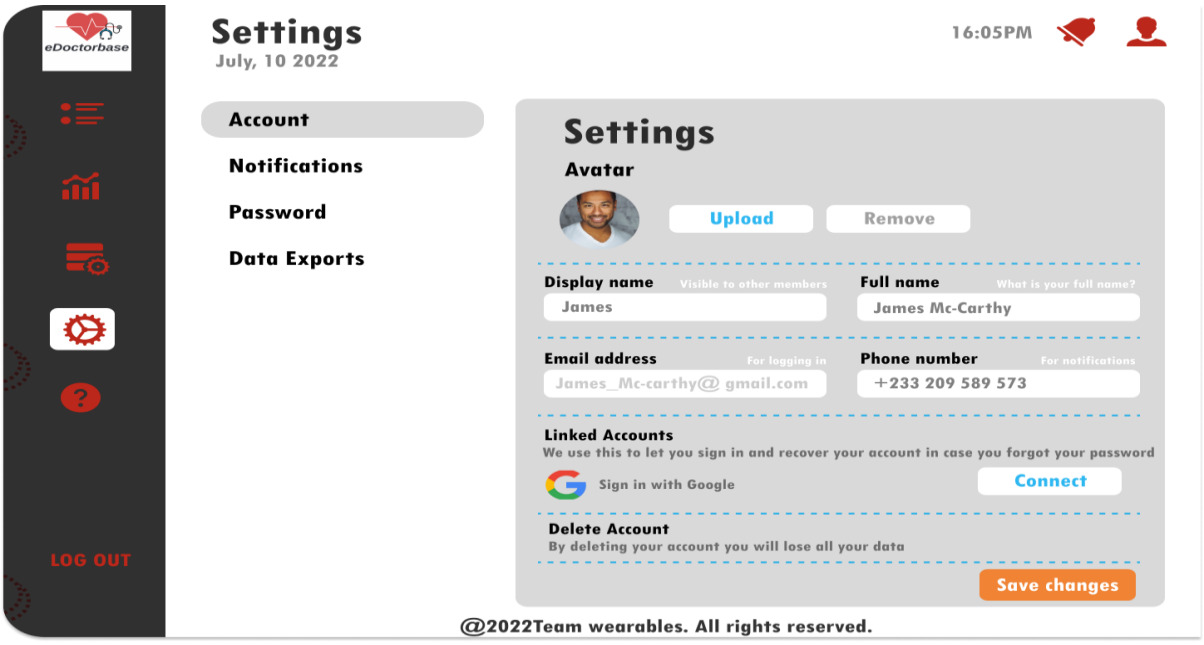


Figure 2.7 Settings menu - *where users can change their name and other login credentials including emails, phone number and many others as shown in the image above.*

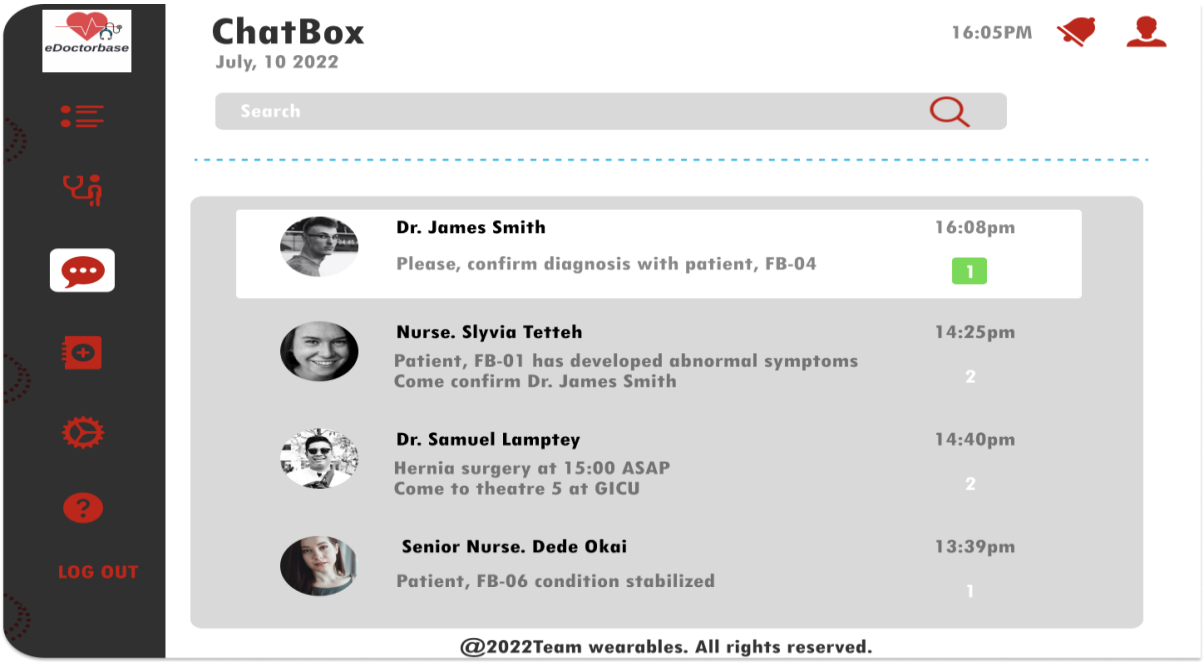


Figure 2.8 Chat box menu - *where users can interact with one another to make information readily available to one other. An example is one Doctor giving another information on a patient he had attended to while the other was away.*

2.2 System development

2.2.1 Front End and UX Development

The Tech Stack for developing the front end of web application was identified and these are:

Hypertext Markup language (HTML 5)

HTML5 is a markup language that is used on the World Wide Web to structure and deliver content [12]. HTML5 contains comprehensive processing models to enable more interoperable design and implementation; it extends, enhances, and validates document markup and introduces markup and application programming interfaces (APIs) for complex web applications, making it a candidate for cross-platform mobile applications [13].

Cascading Style Sheets (CSS 3)

CSS is a style sheet language used to describe the display of a document authored in a markup language such as HTML [14]. CSS is intended to separate display from content, including layout, colors, and fonts [15].

JavaScript (JS)

JavaScript is an ECMAScript-compliant high-level, typically just-in-time compiled language. It supports dynamic typing, prototype-based object-oriented programming, and first-class functions. It supports event-driven, functional, and imperative programming approaches and is multi-paradigm. It supports text, dates, regular expressions, standard data structures, and the Document Object Model via application programming interfaces (APIs) (DOM) [16].

Bootstrap 5

Bootstrap 5 (published in 2021) is the most recent version of Bootstrap (launched in 2013), with additional components, a quicker stylesheet, and increased responsiveness. Bootstrap is a free front-end framework that makes web development faster and easier. It also comes with HTML and CSS-based design templates for typography, forms, buttons, tables, navigation, modals, picture carousels, and many other features, as well as optional JavaScript plugins. Bootstrap also allows you to simply develop responsive designs [17].

Font Awesome

Font Awesome is an icon library and toolkit used by developers on the Internet. It is often inserted as a content delivery network (CDN) link in the HTML page's head section (<head>) and it runs based on JavaScript programming [18].

Chart.js 5

Chart.js is a free, open-source JavaScript data visualization framework that supports eight different chart types: bar, line, area, pie, bubble, radar, polar, and scatter. It is also inserted in the head section (<head>) of an HTML page [19].

The design of the management system in Figma was used as a blueprint to build the user interface (UI) and user experience (UX) of the system. With frameworks mentioned above, front-end functionalities such as the welcome page, sign up, sign in, dashboard, new patient, data visualization charts, manual data upload and the health personnel report page templates was implemented as well [20].

2.2.2 Back End Development

Django 4.1

Django is a Python web framework that promotes rapid development and clean, pragmatic design. It is open source and free [21]. Django follows the MVT design pattern (Model Views

Template) which allows code to be easily written and structured properly. The Model pattern is where data is presented, usually data from a database while the View pattern functions as a request hander and returns the relevant template and content based on the request from the user. The Template pattern contains the layout of the web page (an HTML file), with logic on how to display the data. Django delivers data as an object-oriented mapping (ORM) which makes communication with databases easier without having to write SQL codes [22]

SQLite 3

SQLite is a C-language library that provides a SQL database engine that is tiny, fast, self-contained, high-reliability and full-featured. SQLite is the most widely used database engine on the planet. SQLite is incorporated into all mobile phones and most PCs, as well as innumerable other apps that users use on a daily basis [23].

PostgreSQL

PostgreSQL is a strong object-relational database system that is open source. It has been in active development for almost 15 years and has a proven design that has given it a good reputation for dependability, data integrity, and accuracy [24]. In addition, the user can define his or her own data types, functions, operators, aggregate functions, index methods, and procedural language. It is designed to handle a wide range of workloads, from single computers to big data warehouses or Web applications with many concurrent users [25].

The templates design using the front-end databases was uploaded to our Django project file called ‘myfinalproject’ and using the MVT pattern, python programming was used to implement logic between the templates and the databases. Django’s default database was utilized to create the database tables for the web application, afterwards, the database tables were migrated to PostgreSQL using the postgres server from pgAdmin 4 (The administration platform for PostgreSQL).

Integrated Development Environment (IDE):

Python IDE was run on Microsoft Visual Studio Code (VS Code) version 1.70 and was used for testing the Fitbit API. Django’s environment was also setup and run-on VS CODE. Visual Studio Code is a simplified code editor that supports development tasks such as debugging, task execution, and version control. It seeks to give only the tools a developer requires for a speedy code-build-debug cycle, leaving more complex workflows to a more powerful IDE [26].

Fitbit Web Application Programming Language (API):

Fitbit Web APIs can be used by developers to get Fitbit user data collected by Fitbit trackers and smartwatches, Aria & Aria 2 scales, and manually input log data. Anyone may use the Web APIs to build integrations with Fitbit data services, as long as their application conforms with the Fitbit Platform Terms of Service, the Fitbit Platform Developer and User Data Policy, and the Fitbit user consents to the developer's application [27][28]. Users would have to register a Fitbit account and login to the Fitbit android or IOS mobile application.

Huami Web Application Programming (API):

Huami offers a Web API for accessing user activity data collected by Huami wearable devices. Anyone can create an application to access a user's data on their behalf, as long as the user has given permission [29]. Users would have to be signed in to the Zepp Life android or IOS mobile application

2.3 Testing

Manual Testing

The web application was tested manually by following 5 steps, which are:

- Functionality

The functionality of the web application involved API connectivity, database connections, collection of user information and transiting between web pages.

- Usability

For usability, the overall performance was tested by bringing onboard our peers and colleagues to test our web application and give us honest feedback.

- Interface

The connectivity between the application’s server and the web server was tested. The manual vitals upload page as well as the new patient and data visualization page was tested.

- Compatibility

The management system was tested against different devices and browsers to determine the responsiveness and compatibility.

- Performance

The web application's performance was tested under a heavy load and on different internet speeds.

- Security

The web application was tested to find out whether it did not leak patient information and it was tested for its weak points and vulnerabilities.

2.4 Web hosting and deployment

The web application was hosted on Heroku's server and deployed via Heroku's platform. Heroku is a cloud platform that allows businesses to create, deliver, manage, and scale apps. Heroku simplifies and streamlines the processes of deploying, configuring, scaling, tweaking, and administering apps so that developers can focus on what matters most: creating exceptional products that delight and engage customers. Heroku also supports a wide range of programming languages [30].

For deploying our application in Heroku's cloud services we took 10 steps. These are:

1. Creation of our Django application via a python IDE on VS Code.
2. Installation of Additional packages

   They include:

   - psycopg2

   Psycopg2 is an adapter package required for PostgreSQL database [31]

   - gunicorn

   Gunicorn is a web server gateway interface which Heroku needs to run our application on their platform [32].

   - django-heroku

   Django-heroku is a package that automatically configures a Django application to work on Heroku [33].

   - white noise

   White noise allows web applications to collect static files. Static files are the media files, CSS and JS files used during the development of the Django app [34].
3. Creation of a runtime.txt file

This file basically tells Heroku the version of python that is going to run on the server. Heroku 22 currently supports python-3.10.7 (stable and recommended) and python-3.9.14

4. Creation of a Procfile.

   A Procfile, also known as a Process file is the most important file which states the version and number of workers of a web server gateway interface that Heroku needs to run. Without the Procfile, an application would not run.

5. Generation of a requirements.txt file

A requirements.txt file is basically a list of the packages a developer installed in his Django project which helped him to develop his Django application.

It is generated using the command pip freeze > requirements.txt

6. Creation of a GitHub repository (repo)

A GitHub repository basically stores the files of your application online in GitHub. An account was created on GitHub and a repo was created called ‘myfinalyearproject’. In addition, our application was pushed from the local storage to GitHub.

7. Creation of an application on Heroku.

Heroku cloud services requires a user to sign up and subscribe to a plan. We signed up on Heroku and subscribed to the basic free plan. We then went ahead to create an application called, ‘edoctorbase-app’ on Heroku.

8. Creation of a Database on Heroku

A postgres database was created and certain credentials were provided.

9. Adjustment of settings.py for postgres Database using the credentials was provided.
10. Adjustment of setting.py for static file collection

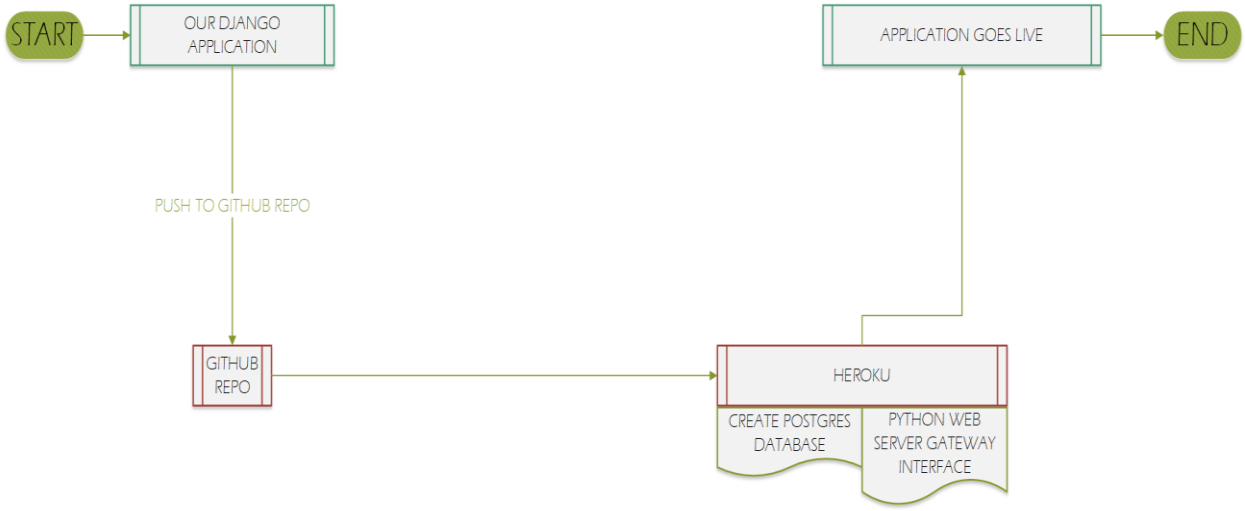


Figure 2.9 Workflow of deploying Django application to Heroku

The figure above shows a workflow from Django application to deployment on Heroku in the right order

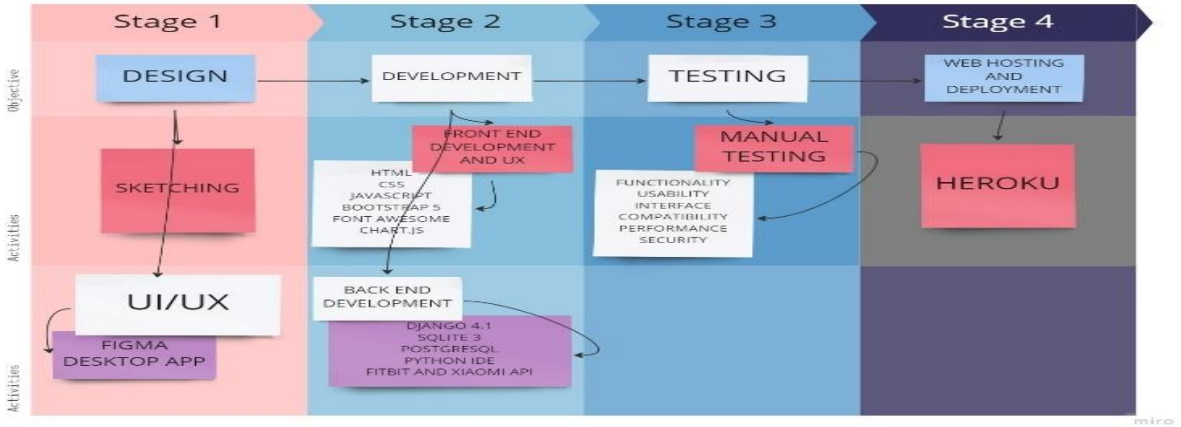


Figure 2.10 Design process

2.5 System Requirements

The requirements for the web application have been grouped into three headings; Safety, Ergonomics and convenience.

SAFETY

- Patient personal information should be saved in a secured database.
- Patient vital signs should be stored in a secured database.

- The application should alert the user when there are abnormal vital sign readings.
- The system should have firewalls.
- The access tokens and Fitbit user ID should be stored in a secured database.

ERGONOMICS

- The user's email used for registration should be valid.
- The user should be able to input the vitals of patients manually and saved into the database.
- The user's password should be alpha numeric and should be 8 or more characters.
- The web application should be responsive (HTML and CSS elements adjust size as the width of the screen and web browser differs)
- The web application should be able to register and save the details of 30 or more patients.
- The web application should be able to communicate with Fitbit API
- The user should be able to input instructions and prescriptions
- The user should be able to register new patients
- The user should be authenticated when signed up and signed in.

CONVENIENCE

- Other users should be able to visualize reports made by different users.
- The user should be able to search for patients using patient code, first name and last name as filters.
- The user should be able to visualize minimum, average and maximum pulse oxygenation and heart rate data easily.
- The user should be able to visualize pulse oxygenation and heart rate data as line charts.
- The application should be aesthetically pleasing.
- The application should be user-friendly
- The application should not have infinite or long-scrolling functionality (data visualization should be on one page for ease of viewing)

Below is an objective tree highlighting all the system requirements

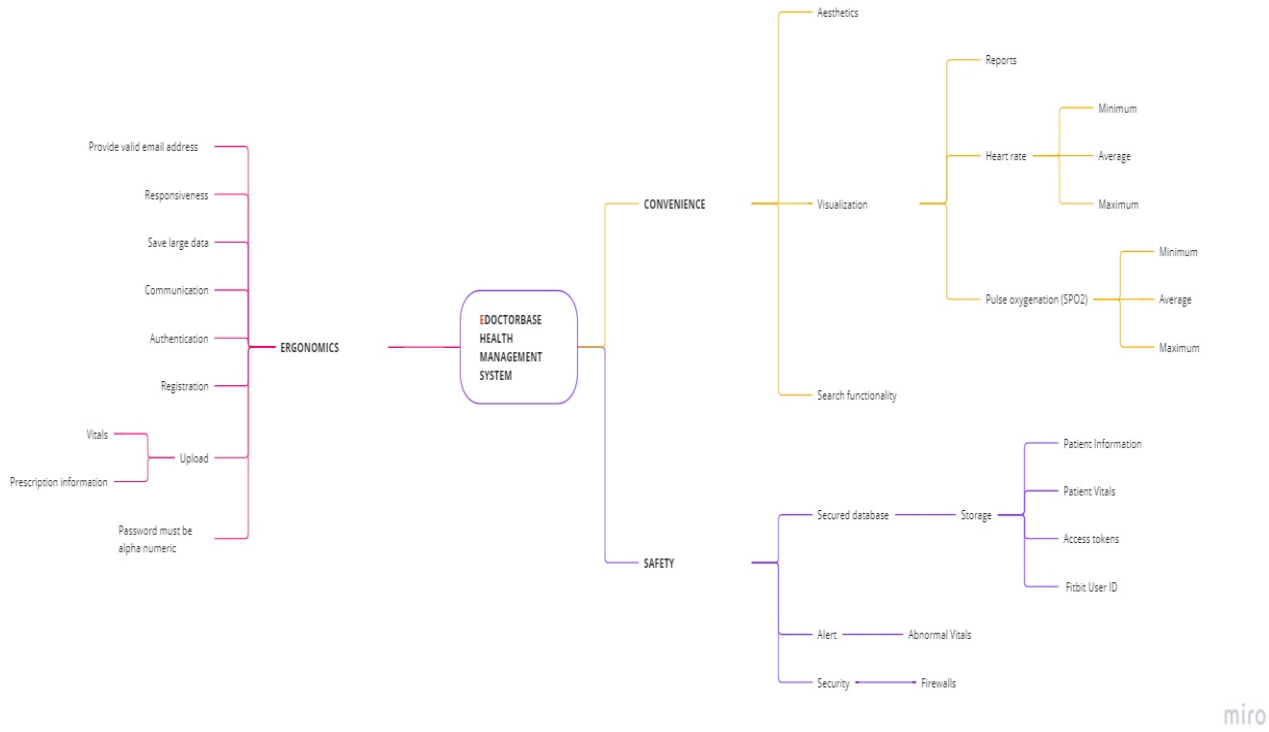


Figure 2.11 Objective tree with the system requirements

2.6 System Specifications

Operating System: Windows 11 professional

Front – End Tools: Figma, HTML5, CSS 3, JavaScript, Bootstrap, Font Awesome, Chart.js

Back-End Tools: Django, SQLite, PostgreSQL

Programming language: Python, JavaScript, HTML CSS, SQL

Integrated Development Environment (IDE): Microsoft Visual Studio Code (VS Code)

Application Programming Interface: Fitbit API and Huami API

Android applications used: Fitbit Web SDK and Zepp Life SDK

REQUIREMENTS	PRIORITY	SPECIFICATIONS	ASSESSMENT
1. Application interface should be user friendly and have good image resolution	9 (High)	1080p 64-Bit computers with stable operating system software	Windows 11 version was found to be the most suitable
2. Application interface should be aesthetically pleasing	8	Text Color, Background Color, Line Charts, Tables, List items, Buttons, Icons, Navigation Bars, Side Bars and Dropdowns	CSS 3, Bootstrap 5, Font Awesome and Chart.js 5 have those functionalities, free and are easy to use.
3. Application content should be easy to design, have consistent formatting and have fluid navigation and action buttons.	8	Paragraph, new line, italics, Bold, alignment, spacing, margins, padding, image and logo placement and adjustments.	HTML5 and JavaScript are the most suitable, free and are easy to use
4. Should be able to save data from the monitoring devices (Have a well secured Database)	9 (High)	>32 Terabytes (32TB)	PostgreSQL has > 32 Terabytes storage size

5. Should support a strong and fast communication module to interact with Fitbit and Xiaomi Servers by means of API	9 (High)	Bluetooth and Wi-Fi Connectivity	Bluetooth 5.0 and Wi-Fi 3.0 was found to be the most suitable
6. SDK APIs should be available and free to download	9 (High)	Free mobile applications in the IOS store and Google play store.	Fitbit SDK and Zepp Life (formally Mi Fit) SDK are available and free to download in the App stores.
7. Bandwidth capacity that either matches or exceeds allowed users	8	At most 10 Concurrent Users Speed -1 Mbps	Heroku's web servers handle concurrent users without causing data traffic
8. Should be able to host data of 30 patients	9 (High)	> 32 Terabytes (32TB)	PostgreSQL can handle > 32 Terabytes of data

9. Back End should be able to interface with Fitbit and Xiaomi Web servers via their APIs	9 (High)	High Level Programming Language	Django, which is a high-level Python web framework and encourages rapid development and clean and well-structured design was found to be the most suitable
10. Should have an environment for writing Python programming language	9 (High)	Python Integrated Development Environment (IDE)	Python IDE and Django's environment can be configured on Microsoft Visual Studio Code. VS Code is free and easy to use.

Table 2.6.1 Design Specifications

2.7 System Design

This includes the architectural diagram of the web application, the operational flow of requesting data from the Fitbit API and Xiaomi API. This section also includes the UML use case and Crow's foot database notation diagrams.

2.7.1 System Architecture

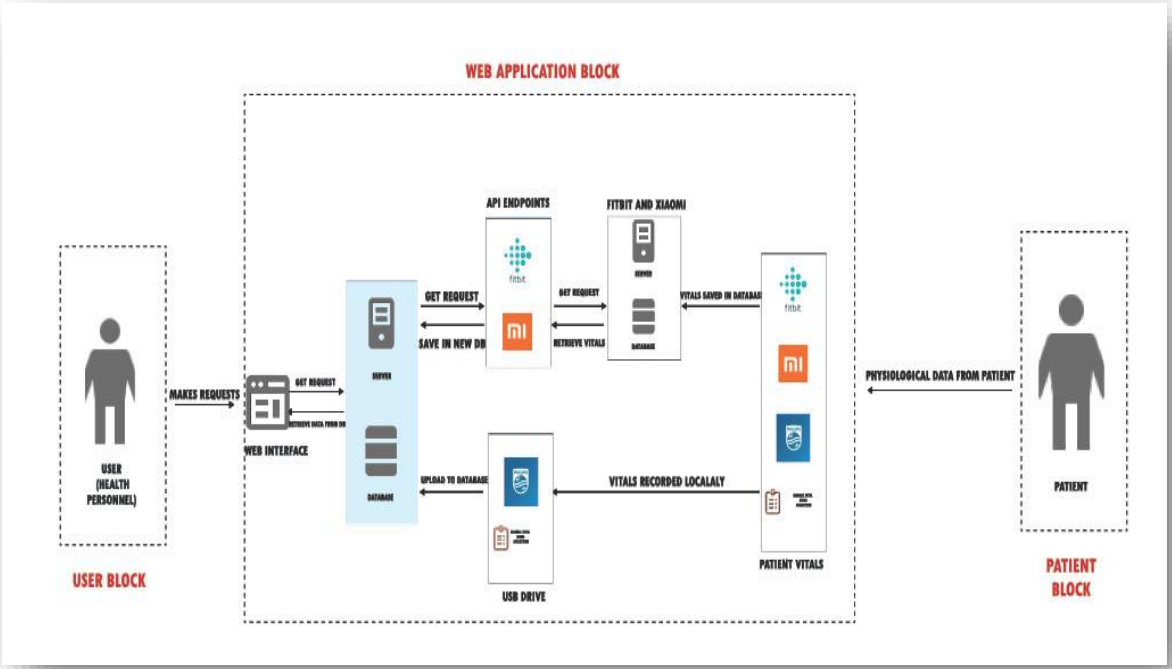


Figure 2.12 Architectural diagram of the web application

A brief description of how the overall system works is given below:

1. When a user submits a request, such as uploading manual vitals, viewing patient data, registering a new patient, viewing reports, searching for a patient, and so on, the web server interacts with the database, retrieving or posting the data requested by the user and displaying it on the web interface for the user to visualize.
2. Fitbit and Xiaomi APIs communicate with their web servers to retrieve physiological data when the user makes a request to visualize patient heart rate and pulse oxygenation data.
3. Patients are registered on the Fitbit and Zepp Life apps using an email and password with the wearables worn on the wrist.
4. Physiological data obtained from patients is stored on cloud databases via the android application web servers.
5. Physiological data is synced every 15 minutes to save data into the cloud databases.
6. Data obtained from manual monitoring is transcribed in an Excel sheet and saved as CSV.

7. Data obtained from the PHILIPS MX 450 Intellivue MX 450 patient monitor is recorded with a USB drive and saved as a CSV file.
8. Data obtained from both the manual monitoring and the PHILIPS MX 450 patient monitor can be uploaded into the web application's database.

2.7.2 System Database Design (Crow's Foot Database Notation)

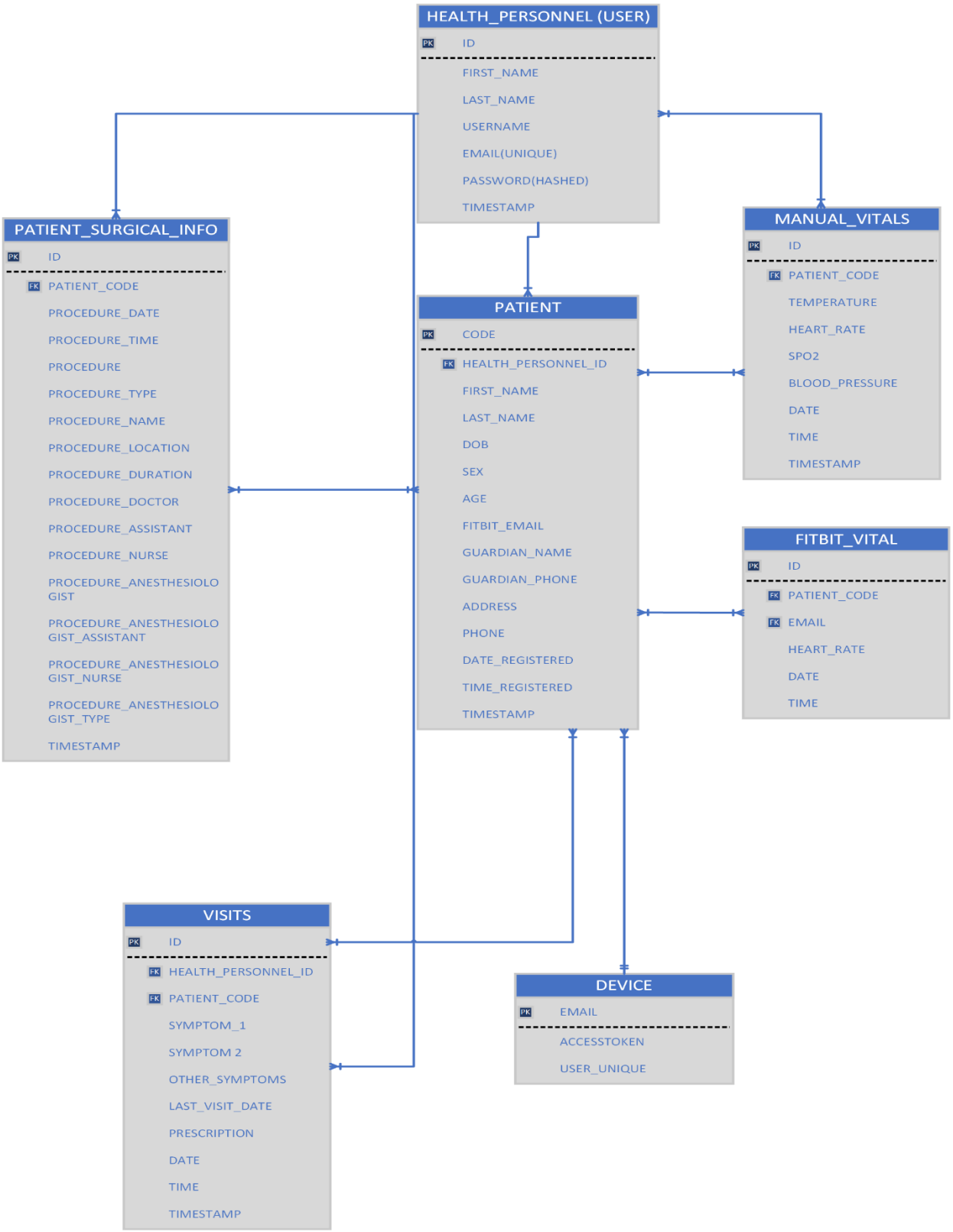


Figure 2.13 Database design of the web application

The data dictionary below explains the crow's foot database notation

Table 2.7.1 shows the data dictionary for the User

ENTITY: HEALTH_PERSONNEL (USER)				
Primary Key	*ID*			
ATTRIBUTES				
NAME	***PRIMARY KEY***	***DATA TYPE***	***NULL***	***COMMENT***
ID	YES	INT	NO	Default (AutoField)
FIRST_NAME	NO	CHAR	NO	Users first name
LAST_NAME	NO	CHAR	NO	Users Last name
USERNAME	NO	CHAR	NO	Username of User. Includes designation
EMAIL	NO	EMAIL	NO	User's email. Email is unique to user
PASSWORD	NO	CHAR	NO	User's password. Password is hashed and alpha numeric
TIMSTAMP	NO	DATETIME	NO	Date and time User created his account.

Table 2.7.2 shows the data dictionary for Manual Vitals collected

ENTITY: MANUAL_VITALS	
Primary Key	*ID*
ATTRIBUTES	

NAME	*PRIMARY KEY*	*FOREIGN KEY*	*DATA TYPE*	*NULL*	*COMMENT*
ID	YES	NO	INT	NO	Default (AutoField)
PATIENT_CODE	NO	YES	CHAR	NO	Patient code given by hospital
TEMPERATURE	NO	NO	INT	NO	Patient's temperature
HEART_RATE	NO	NO	POSITIVE INT	NO	Patient's heart rate
SPO2	NO	NO	POSITIVE INT	NO	Patient's pulse oxygenation
BLOOD_PRESSURE	NO	NO	POSTIVE INT	YES	Patient's blood pressure
DATE	NO	NO	DATE	NO	Date the vitals were taken
TIME	NO	NO	TIME	NO	Time the vitals were taken
TIMSTAMP	NO	NO	DATETIME	NO	Date and time User saved the vitals

Table 2.7.3 shows the data dictionary for the data collected from Fitbit Versa 2

ENTITY: FITBIT_VITAL	
Primary Key	*ID*
ATTRIBUTES	

NAME	*PRIMARY KEY*	*FOREIGN KEY*	*DATA TYPE*	*NULL*	*COMMENT*
ID	YES	NO	INT	NO	Default (AutoField)
PATIENT_CODE	NO	YES	CHAR	NO	Patient code given by hospital
EMAIL	NO	YES	EMAIL	NO	Fitbit Email with access token and user ID assigned to patient
HEART_RATE	NO	NO	POSITIVE INT	NO	Patient's heart rate
DATE	NO	NO	DATE	NO	Date the vitals were taken
TIME	N0	NO	DATE		Time the vitals were taken

Table 2.7.4 shows the data dictionary for the parameters required for the Fitbit API

ENTITY: DEVICE				
Primary Key	*EMAIL*			
ATTRIBUTES				
NAME	*PRIMARY KEY*	*DATA TYPE*	*NULL*	*COMMENT*
EMAIL	YES	EMAIL	NO	The primary for this entity

ACCESSTOKEN	NO	CHAR	NO	Access token obtained from Fitbit API
UNSER_UNIQUE	NO	CHAR	NO	User Id obtained from Fitbit API

Table 2.7.5 shows the data dictionary for Patient's report

ENTITY: VISITS					
Primary Key	*ID*				
ATTRIBUTES					
NAME	***PRIMARY KEY***	***FOREIGN KEY***	***DATA TYPE***	***NULL***	***COMMENT***
ID	YES	NO	INT	NO	Default (AutoField)
HEALTH_PERSONNEL	NO	YES	CHAR	NO	User's username saved when registered
PATIENT_CODE	NO	YES	CHAR	NO	Patient code given by hospital
SYMPTOM_1	NO	NO	CHAR	NO	Patient's symptoms
SYMPTOM_2	NO	NO	CHAR	NO	Patient's symptoms
OTHER_SYMPTOMS			CHAR		Patient's symptoms
PRESCRIPTION	NO	NO	TEXT	NO	Prescription instruction

					written be user to be seen by other users
DATE	NO	NO	DATE	NO	Date the patient visited the hospital
TIME	N0	NO	DATE	NO	Time the patient visited the hospital
TIMESTAMP	NO	NO	DATETIME	NO	Date and Time the User submitted the form

Table 2.7.6 shows the data dictionary for patient surgical information

ENTITY: PATIENT_SURGICAL_INFORMATION					
Primary Key	*ID*				
ATTRIBUTES					
NAME	***PRIMARY KEY***	***FOREIGN KEY***	***DATA TYPE***	***NULL***	***COMMENT***
ID	YES	NO	INT	NO	Default (Auto Field)
PATIENT_CODE	NO	YES	CHAR	NO	Patient code given by hospital
PROCEDURE_DATE	NO	NO	CHAR	NO	Date of surgery

PROCEDURE_TIME	NO	NO	CHAR	NO	Time of surgery
PROCEDURE	NO	NO	TEXT	NO	How the surgery would be done
PROCEDURE_TYPE	NO	NO	CHAR	YES	Type of surgery being done
PROCEDURE_NAME	NO	NO	CHAR	YES	Surgery name
PROCEDURE_LOCATION	N0	NO	CHAR	YES	Surgery location
PROCEDURE_DURATION	NO	NO	CHAR	YES	Surgery duration
PROCEDURE_DOCTOR	NO	NO	CHAR	NO	Surgeon taking charge of the surgery
PROCEDURE_ASSISTANT	NO	NO	CHAR	NO	Surgeon's assistant
PROCEDURE_NURSE	NO	NO	CHAR	YES	Nurse in charge of surgery
PROCEDURE_ANESTHESIOLOGIST	NO	NO	CHAR	YES	Anesthesiologist in charge of surgery
PROCEDURE_ANESTHESIOLOGIST_ASSISTANT	NO	NO	CHAR	YES	Anesthesiologist's assistant
PROCEDURE_ANESTHESIOLOGIST_NURSE	NO	NO	CHAR	YES	Anesthesiologist's nurse

PROCEDURE_ANESTHESIOLOGIST_TYPE	NO	NO	CHAR	YES	Anesthesiologist type
TIMESTAMP	NO	NO	DATETIME	NO	Date and Time the User submitted the form

Table 2.7.7 shows the data dictionary for Patient Information

ENTITY: PATIENT					
Primary Key	*PATIENT_CODE*				
ATTRIBUTES					
NAME	***PRIMARY KEY***	***FOREIGN KEY***	***DATA TYPE***	***NULL***	***COMMENT***
PATIENT_CODE	YES	NO	CHAR	NO	Patient code given by hospital
FIRST_NAME	NO	NO	CHAR	NO	Patient first name
LAST_NAME	NO	NO	CHAR	NO	Patient Last name
DOB	NO	NO	DATE	NO	Patient Date of birth
SEX	NO	NO	CHAR	NO	Patient Gender
AGE	NO	NO	POSITIVE INT	NO	Patient Age
FITBIT_EMAIL	NO	YES	EMAIL	NO	Email from Device entity that is assigned to the patient
GUARDIAN_NAME	NO	NO	CHAR	NO	Full name of patient guardian
GUARDIAN_PHONE	NO	NO	CHAR	NO	Contact of patient's guardian
ADDRESS	NO	NO	CHAR	NO	Residential address of patient. Can

					be a digital address
PHONE	NO	NO	CHAR	YES	Contact of patient
OCCUPATION	NO	NO	CHAR	YES	Occupation of patient
DATE_REGISTERED	NO	NO	DATE	NO	Date patient was registered
TIME_REGISTERED	NO	NO	TIME	NO	Time patient was registered
TIMSTAMP	NO		DATETIME	NO	Date and time User submitted the form

2.7.3 UML USE CASE DESIGN

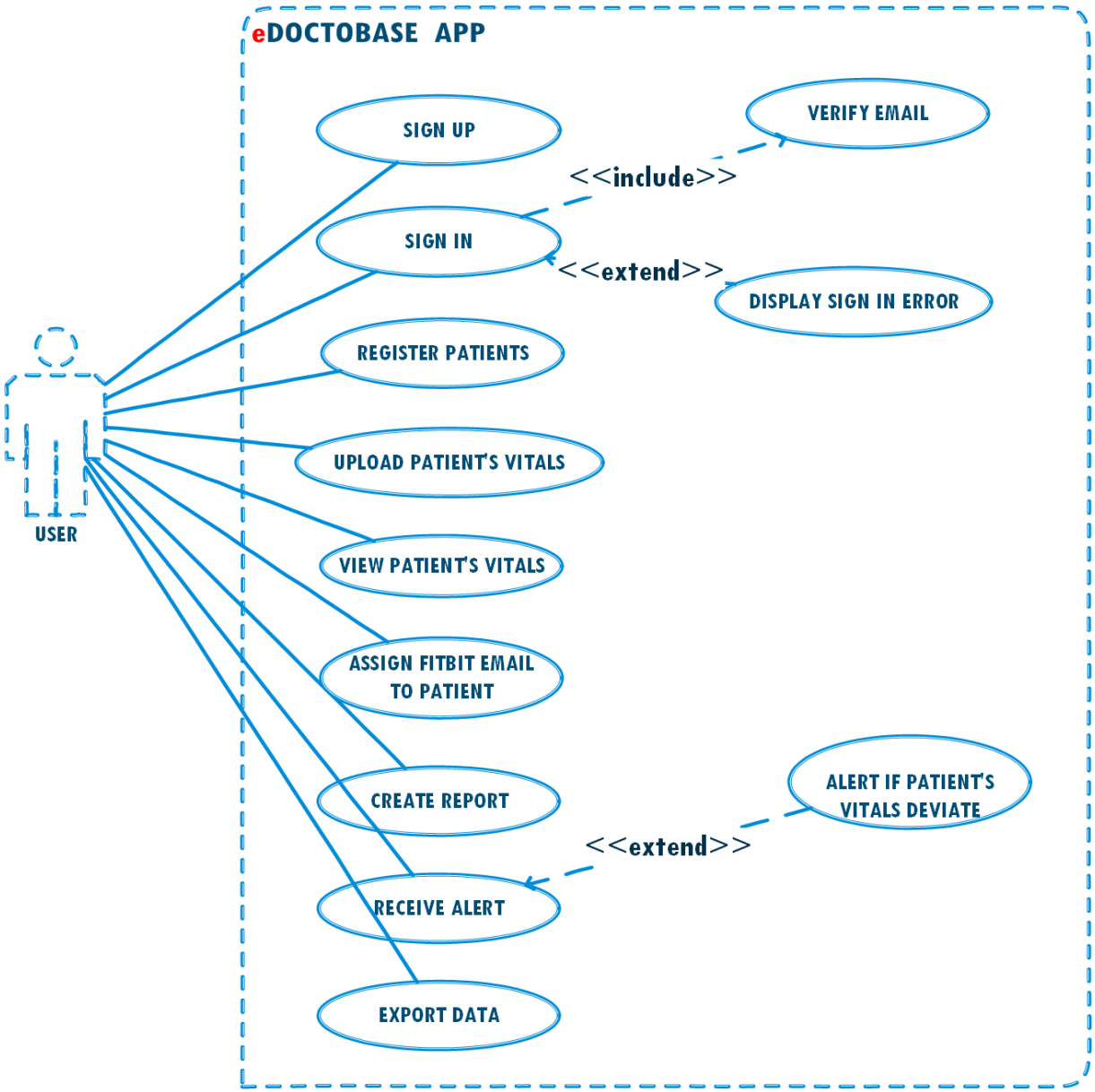


Figure 2.14 UML use case diagram of the web application

The summary of the actions the user is able to perform is given in the table below:

Table 2.7.8 shows the actions a user can perform.

ACTIONS	USER	OTHER INFORMATION
SIGN UP	Create an account and validates account	Has to verify email to have access to the web application
SIGN IN	Require username and password	Displays sign in error when credentials are invalid
REGISTER NEW PATIENTS	Input patient information and surgical information (if applicable) and save in database	
UPLOAD PATIENT VITALS	Upload vital signs results when taken manually.	Usually taken every 4 hours.
VIEW PATIENT VITALS	Heart rate data retrieved from Fitbit API.	
ASSIGN FITBIT EMAIL TO PATIENT	Can assign different Fitbit emails to patients so that data can be retrieved from that email address.	
CREATE REPORT	Includes giving prescription information and recording patient symptoms.	Different clinicians can view patient reports as well.
RECEIVE ALERT	Colour change	Occurs when patient vitals are abnormal
EXPORT DATA	Can import and Export patient's information and vitals	Data export tool

2.7.4 Flowchart For Fitbit API

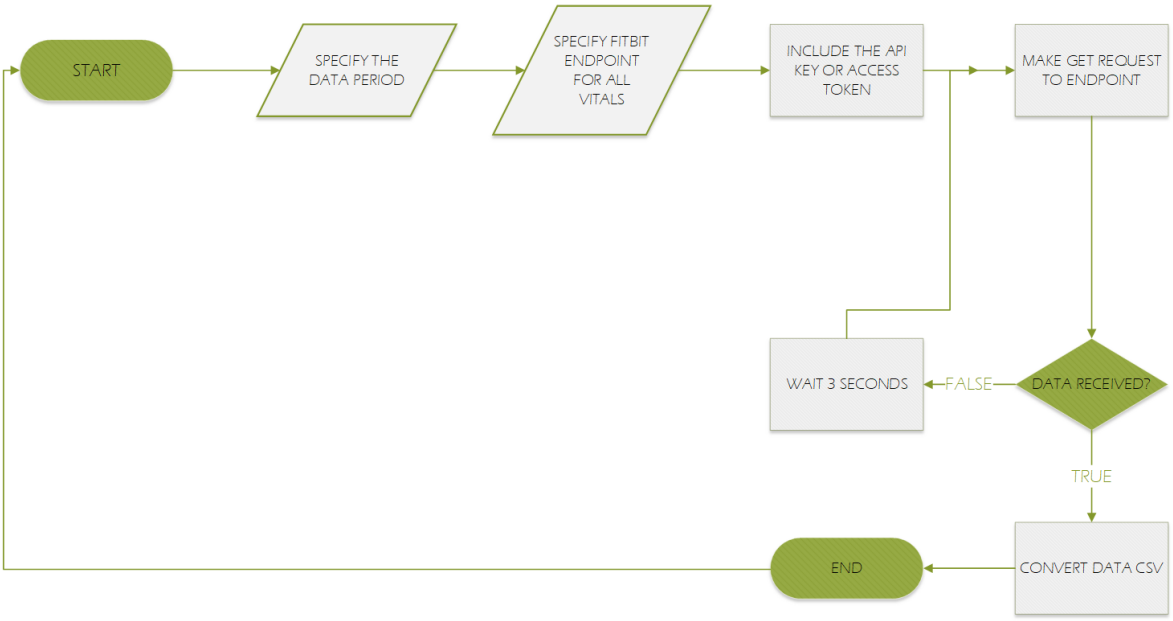


Figure 2.15 Flowchart chat for requesting data from Fitbit Versa 2

The process starts with registering a Web API in the Fitbit API web page (https://dev.fitbit.com/) with an email. The period for which the data extraction is required is now specified, the Fitbit API is authorized and an access token and user ID is then generated for the registered Fitbit API. However, the access token and user ID is specific to an email address.

End points are found in APIs. They represent the exact data a user wants to make a request for. A login endpoint and a signup endpoint are some examples. They make the API aware of what type of request the user wants to make. Using any programming language of your choice, the Fitbit endpoints are then specified for temperature, heart rate and SpO2, and the access token and user ID generated earlier is included in your code.

The code is then run (can be run in a terminal or a separate window) and when code execution is successful (true), the data is converted to CSV file format. However, when there is an error, such as; network errors, heavy traffic to the endpoint, access token expiration, the executable returns as false and waits for 3 seconds before it makes a request to the endpoint again. The data is then converted to CSV file format when it returns as true. Otherwise, it will run in a loop until the error(s) has been rectified.

2.7.5 Flowchart for Xiaomi (Huami API)

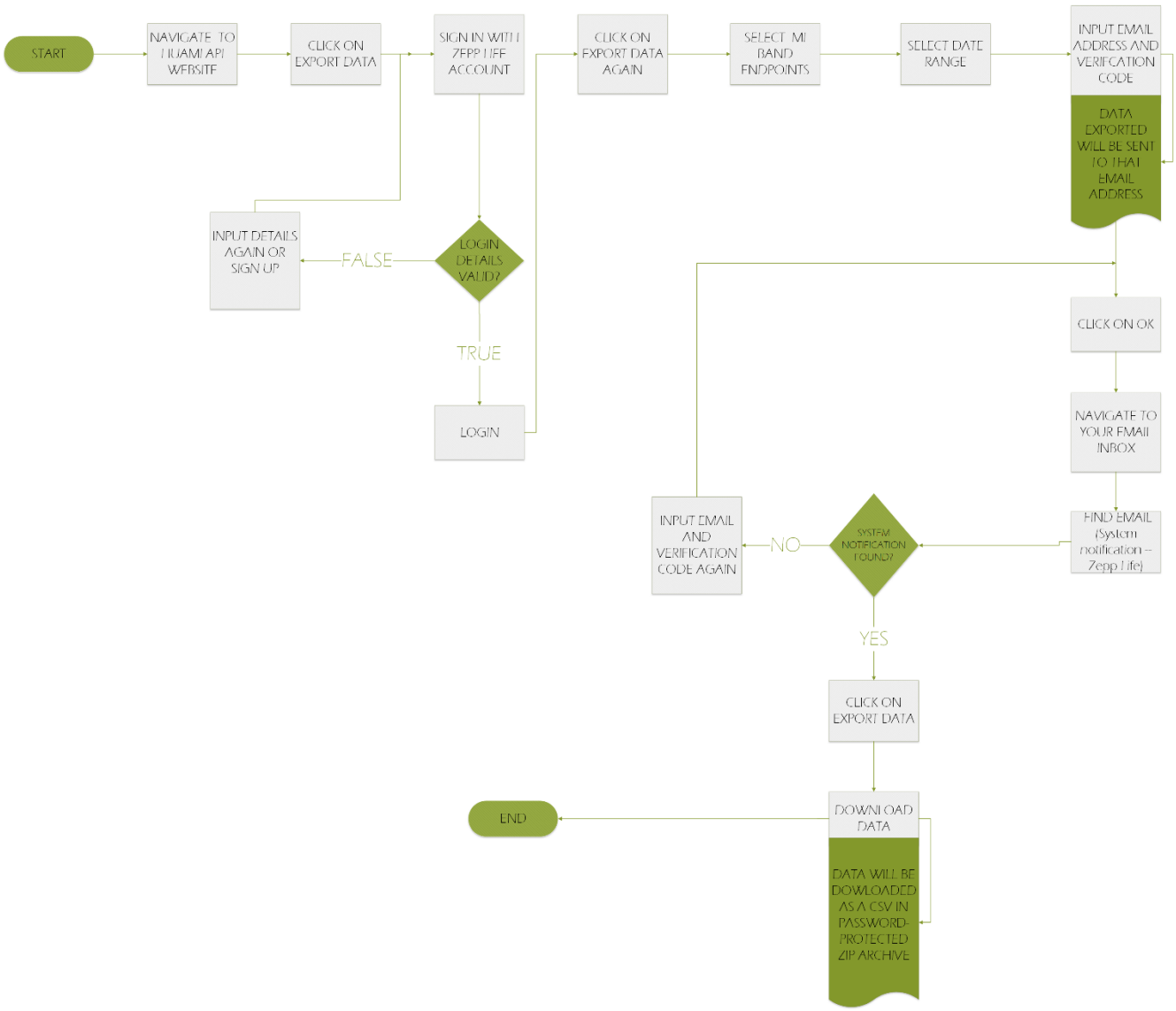


Figure 2.16 Flowchart for requesting data from Xiaomi Mi band 6

The user visits the Huami API web page (mifit.huami.com/t/account mifit) and selects the export data option. The user then logs in with his Zepp Life application credentials (or establishes a new account if he does not already have one). The user then clicks on export data once more and specifies the desired end points and date range.

Huami asks the user to enter an email address and a verification code in order for the data to be sent to that address.

The user then goes to his email inbox and looks for the Zepp Life message. Finally, the user selects export data and downloads the data as a CSV file in a password-protected zip archive. Huami provides the password, and the user extracts the zip file to retrieve the data.

2.8 System Functionality and Implementation

2.8.1 Introduction

From the UML use case diagram, users would mainly be able to perform the following actions:

Sign up, Sign in, Register new patients, Upload patient vitals, View patient vitals, assign Fitbit email to patient, create report, receive alerts and export data.

2.8.2 Sign Up

The user would be required to provide the following credentials: first name, last name, username, designation, email, and password to sign up to the system. A verification link would be sent to the email provided earlier and the user would be required to click on the link in order to have access to the system.

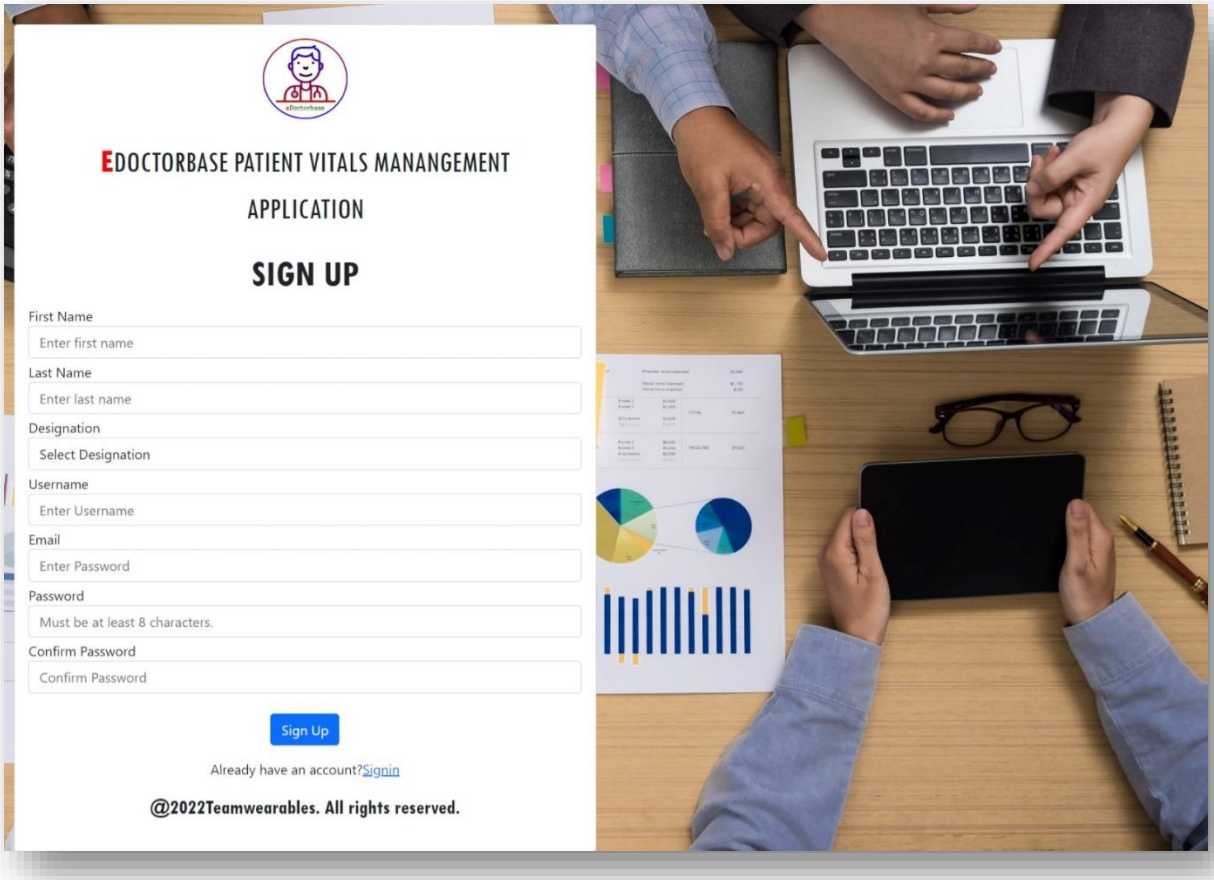


Figure 2.17 System's sign-up page

2.8.3 Sign In

The user would be required to provide his username and password as credentials to sign in to the system. When the email and password provided is valid, the system allows the user to have access to the system.

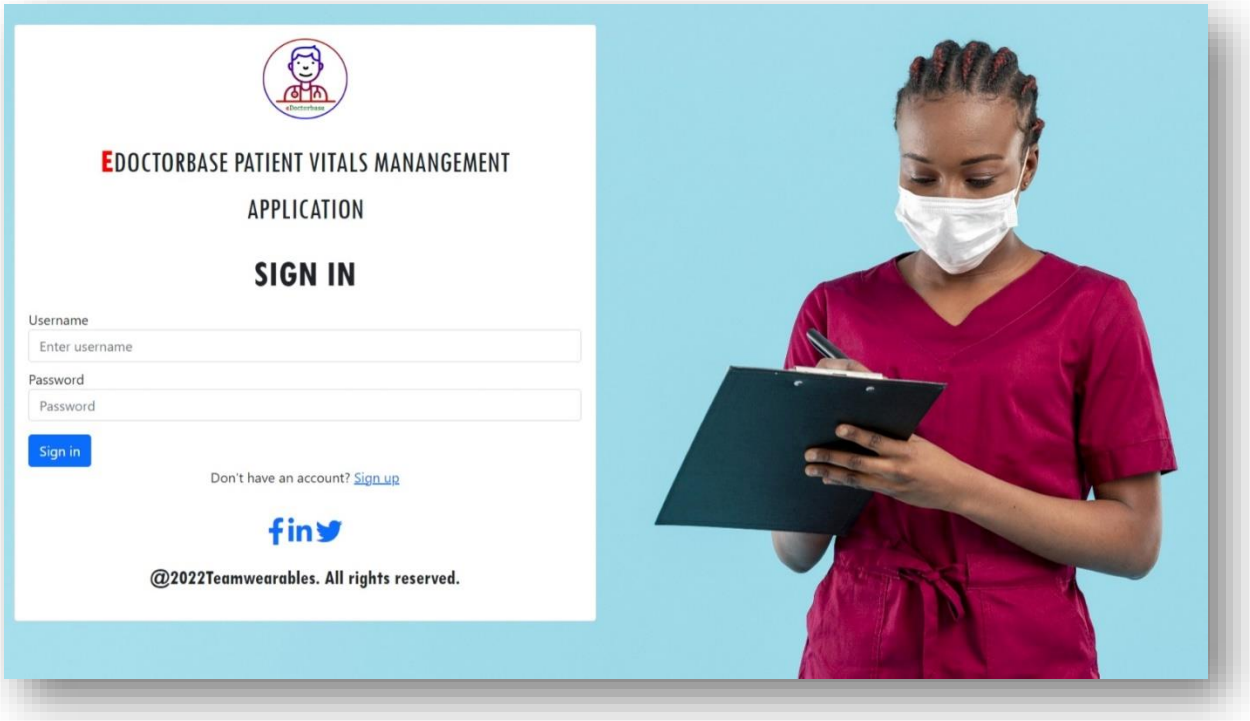


Figure 2.18 System's sign in page

2.8.4 Register New Patient

The user is able to register a patient if it is the patient's first time of visiting the hospital. The patient is given a unique Outpatients Department (OPD) number, which is used for identification. This number would be used to identify each patient anytime he or she visits the hospital. In addition, the user would be able to save the patient's surgical history in the system's database. This helps to maintain the medical history of the patient.

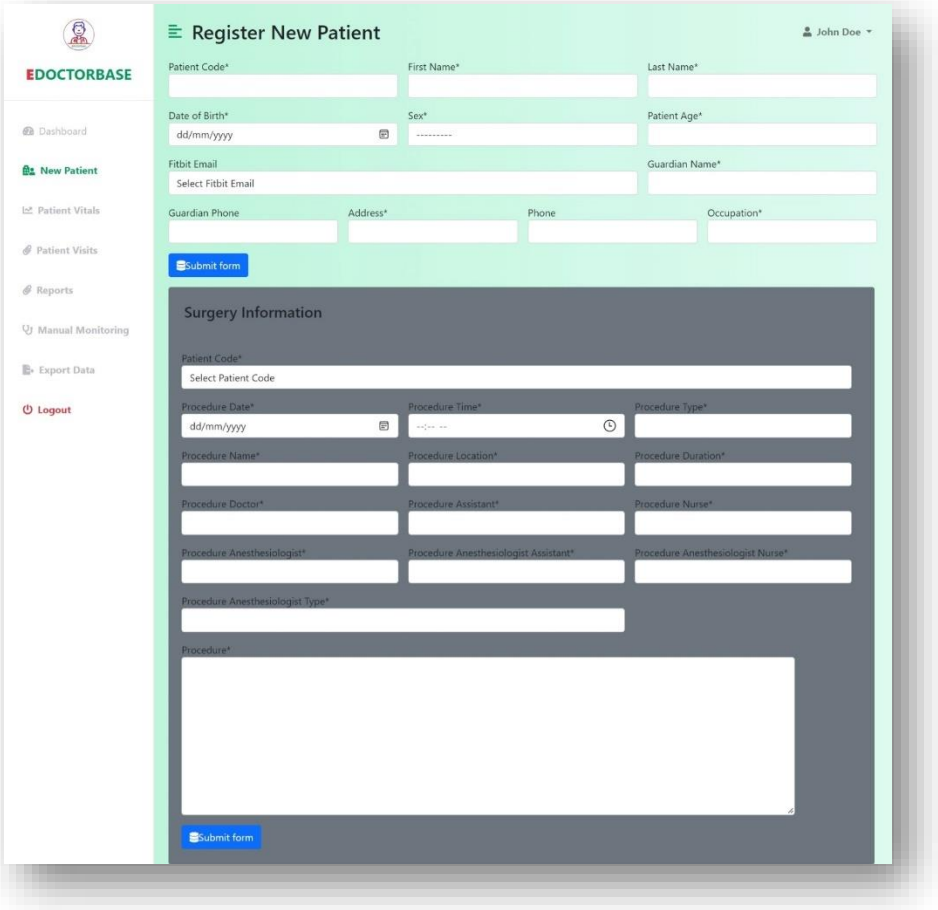


Figure 2.19 New patient registration and surgical information page

2.8.5 Upload Patient Vitals

Vitals signs are taken in the hospital every 4 hours and written in a document for safekeeping and records. The user would however, be able to upload patient vitals into the systems database that is a more secure way of storage. In addition, the vitals data can be retrieved as a csv file whenever it is needed.

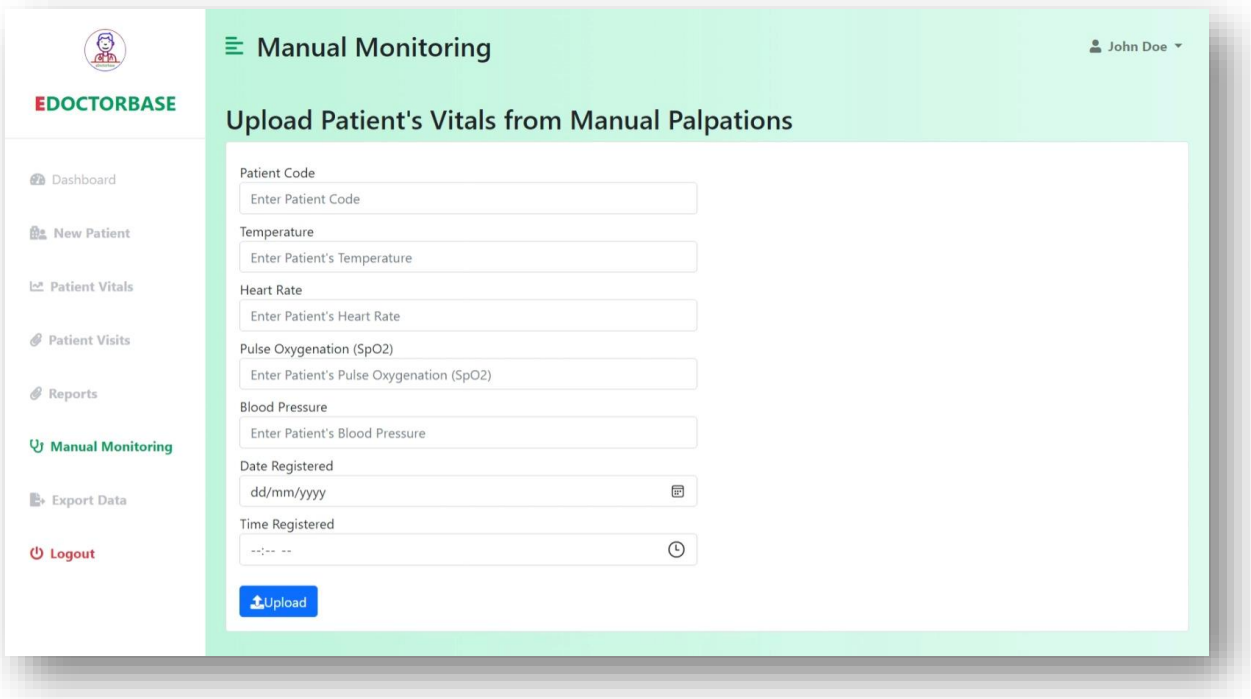


Figure 2.20 Patient vitals upload page

2.8.6 View Patient Vitals

Patient heart rate obtained from Fitbit web API is displayed here as a line chart. Manual pulse oxygenation (SpO2) data is also displayed as a line chart. The chart displays minimum, average and maximum heart rate and SpO2 trends. A table included also shows minimum, average and maximum heart rate and SpO2 for the last 24 hours, 4 hours, 1 hour and 5 minutes respectively.

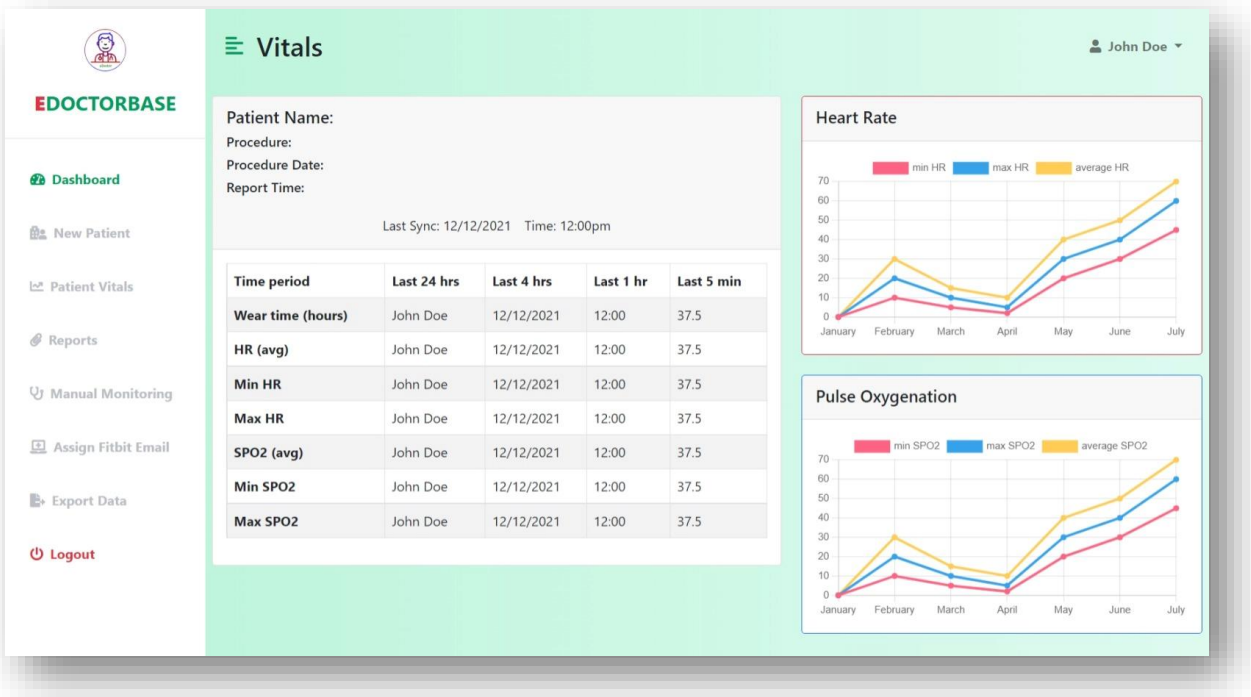


Figure 2.21 Patient physiological data visualization page

2.8.7 Receive Alerts for Normal Vitals

The table included in the vitals visualization page alerts the user of abnormal heart rate and SpO2 data by changing the colour to red when vitals are high and blue when vitals are relatively low.

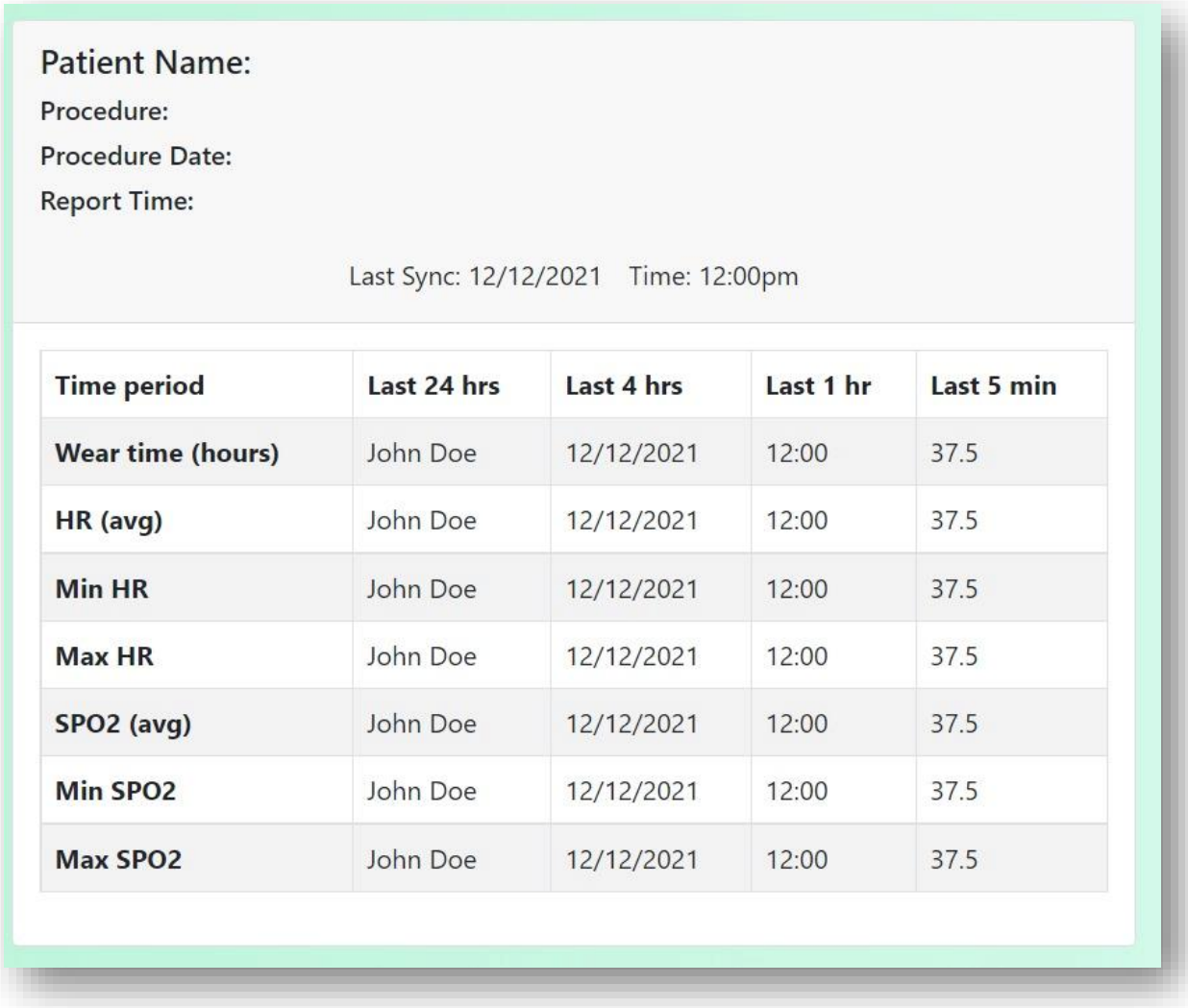


Figure 2.22 Table that is an indicator of abnormal vitals

2.8.8 Assign Fitbit Email To Patient

Fitbit emails are saved on the database together with access token and user IDs that can be used to get patient heart rate data via Fitbit Web APIs. Fitbit emails are then assigned to patients when registering to view vital signs and assigned to an invalid Fitbit email when patient vitals are not being monitored. The reason is that Fitbit emails are specific to one Fitbit user, therefore Fitbit email can be assigned to one (1) patient at a time. This functionality is to ensure that many patients can have their vitals taken by the same wearable device at separate instances of wearing the device.

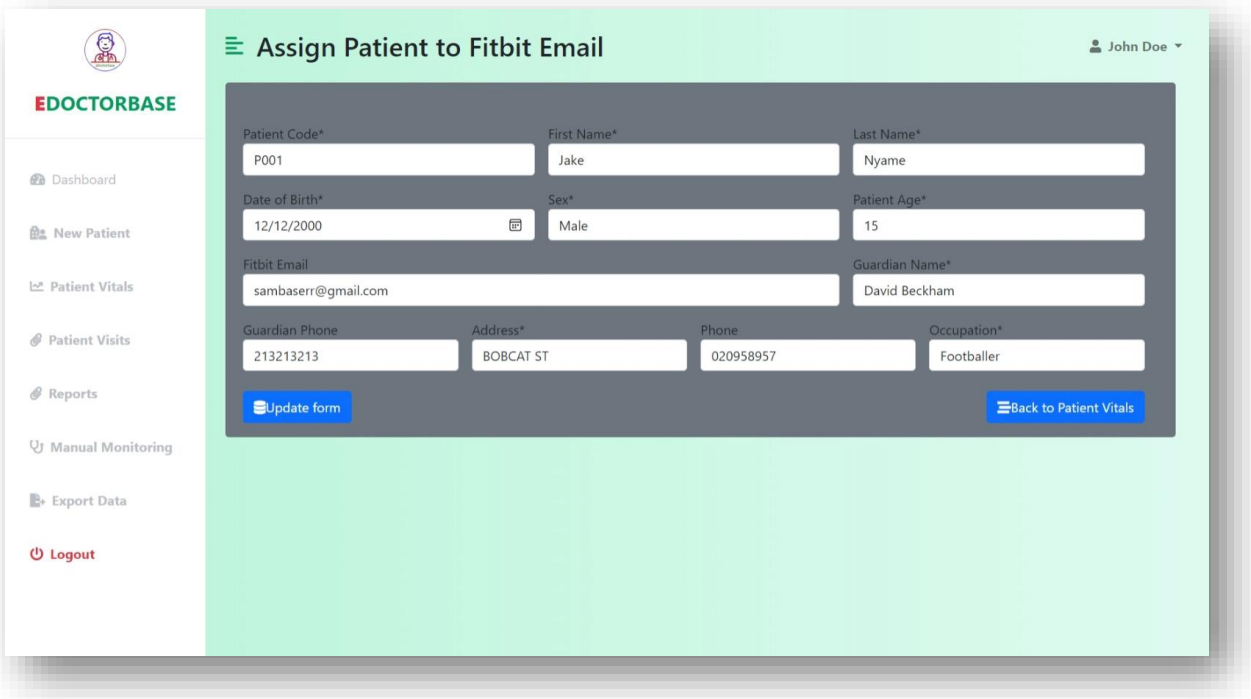


Figure 2.23 Assign Fitbit email to patient

2.8.9 Create Report

Whenever a patient visits a hospital, the patient complains to the medical professional and the medical professional prescribes drugs and relief items for the patient. This is the idea behind this functionality. The user can input the patient's symptoms and write out prescriptions for the patient. Other users can also view the instructions or prescriptions from the initial user. This promotes accountability and improves the quality of health of patients.

Figure 2. 24 Report form

2.8.10 Export Data

Users would be able to export all patient's information as a CSV file. In addition, they would be able to export all patient surgical information, Fitbit data and manual monitoring data as CSV files. Also, users would be able to export each patient's personal information, surgical information, Fitbit data and manual monitoring data as CSV files. Users would also be able to import data into the system's database.

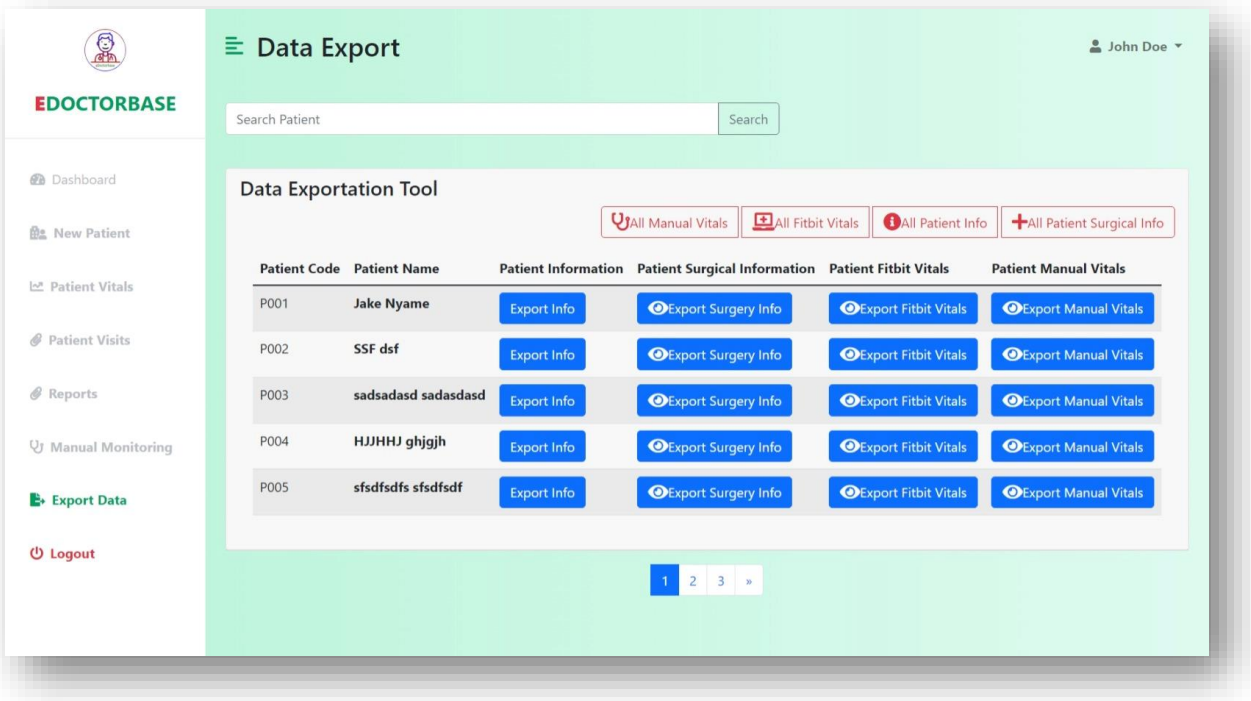


Figure 2.25 Data export page

2.9 Other Functionality Implementation

2.9.1 Welcome Page

Users would be greeted by a welcome page when they input the URL of the system.

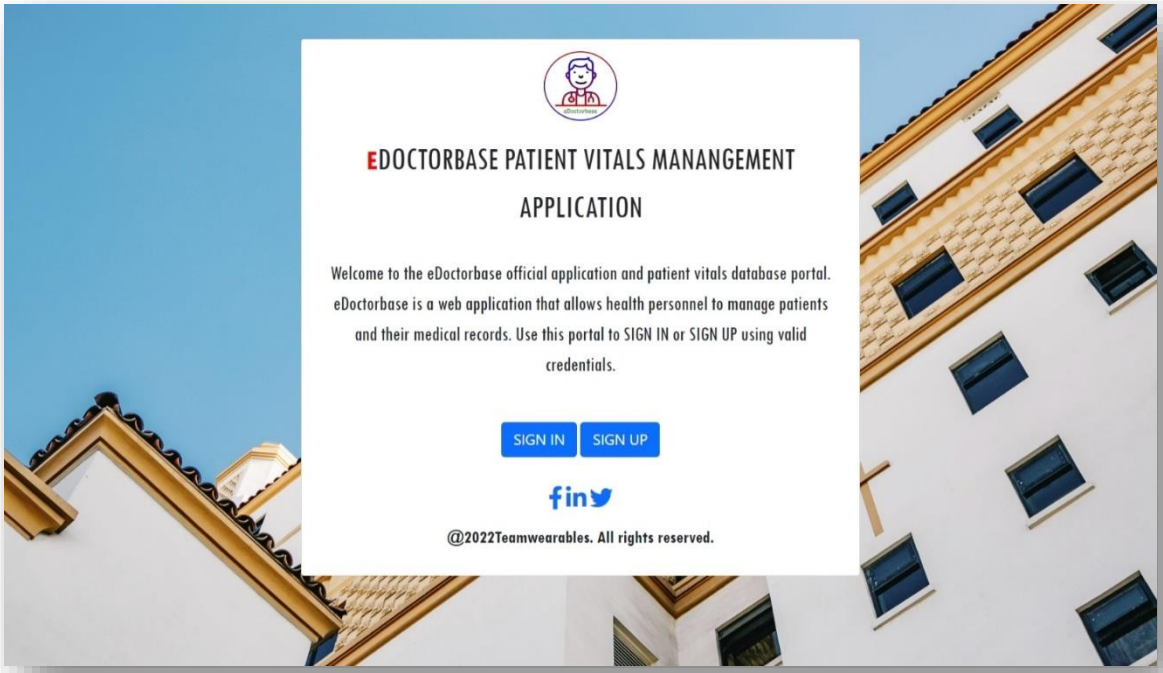


Figure 2.26 Welcome page

2.9.2 Dashboard

Users would be able to view recent patient information such as heart rate, SpO2, temperature and step count.

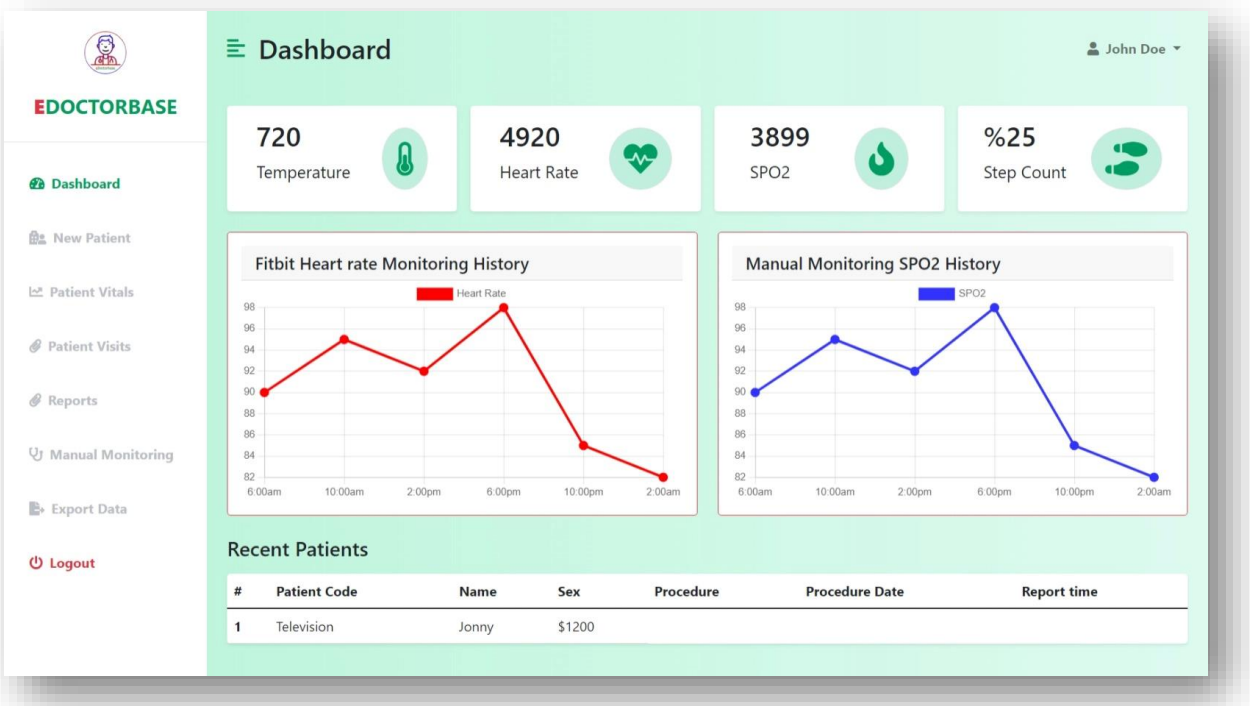


Figure 2.27 User's dashboard

Chapter3 Web Application Testing

3.1 Administrator Panel

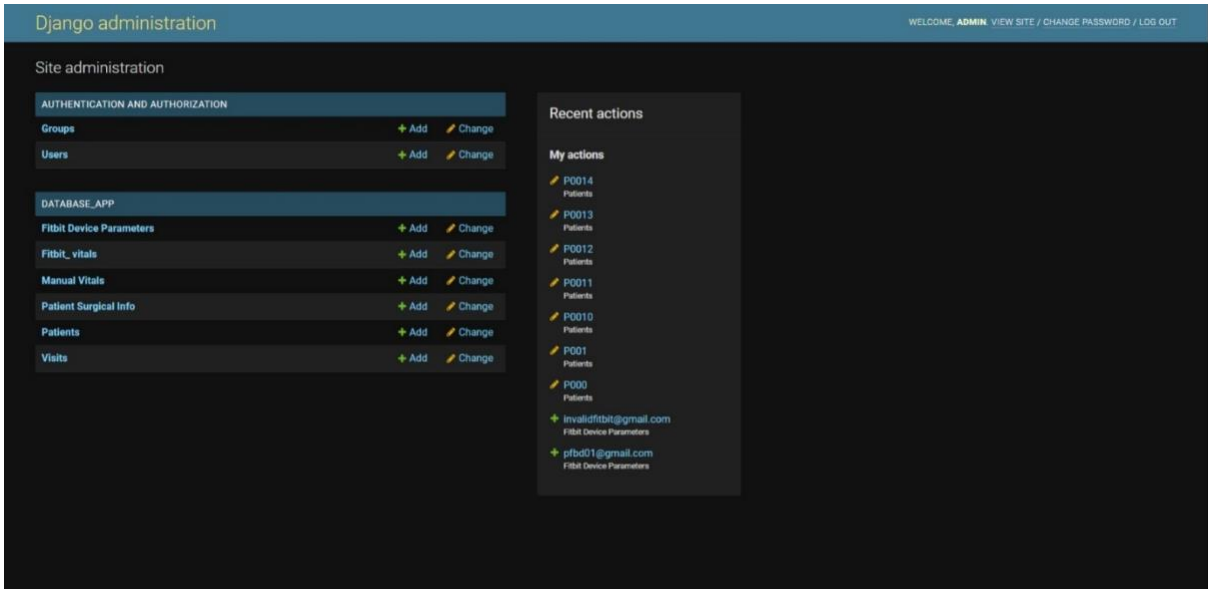


Figure 3.1 Database tables

3.2 Sign Up Validation

A message would be sent from the admin's email to the registered user email, providing the user's login details so when he forgets he can easily refer to his email to get the details. Also, an email is sent for the user to confirm his account.

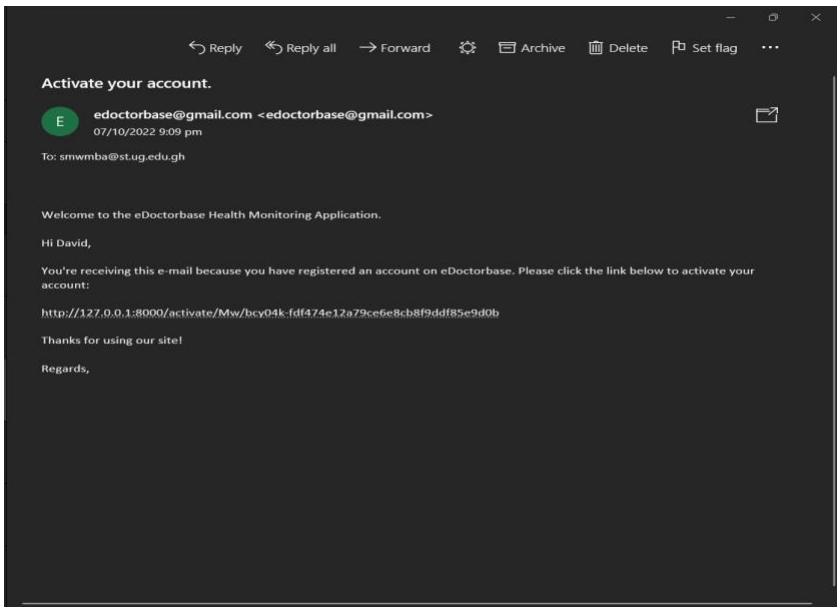


Figure 3.2 Account confirmation email -*shows a message sent from the admin's email to the registered user email, asking the user to confirm his account*

3.3 Sign In Validation

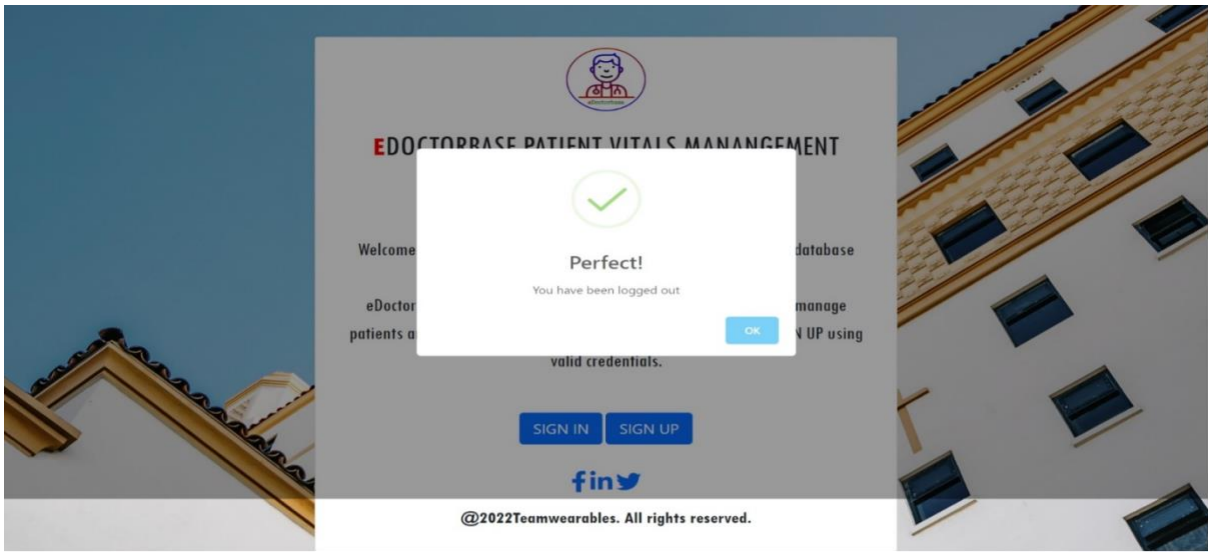


Figure 3.3 Dashboard of a registered user

3.4 Logout Functionality

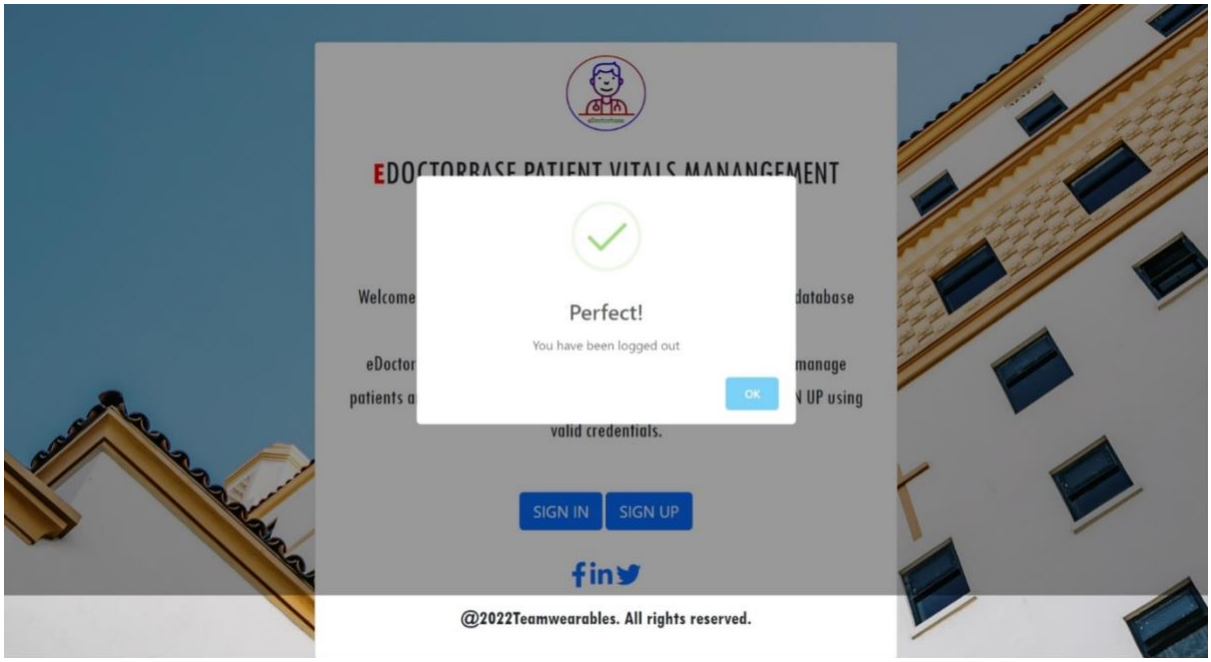


Figure 3.4 Successful Page Logout

3.5 Dashboard of a Registered User

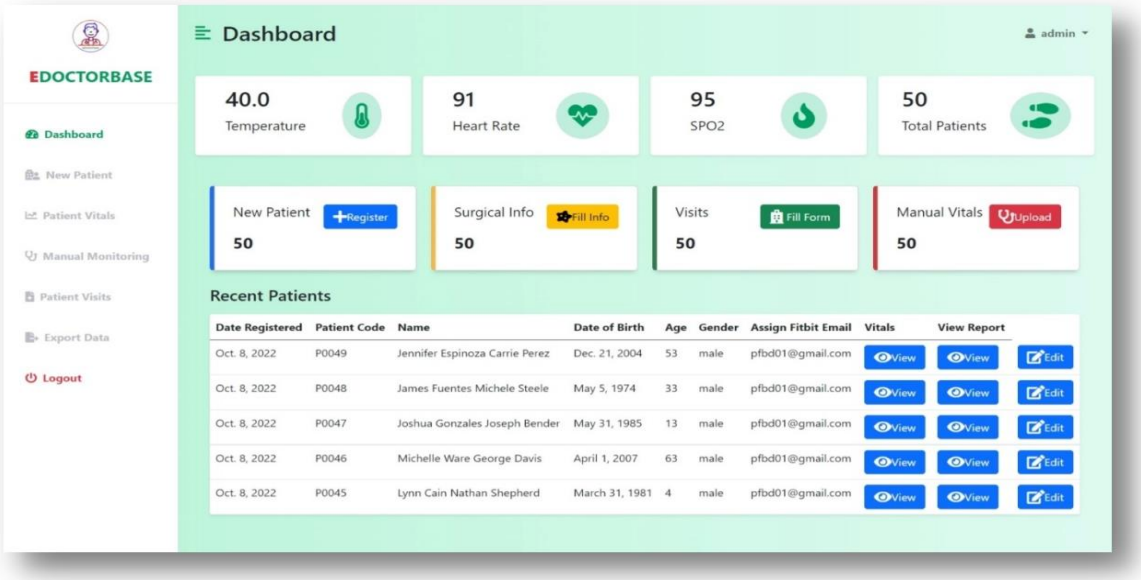


Figure 3.5 Dashboard of the admin

3.6 Register New Patient Functionality

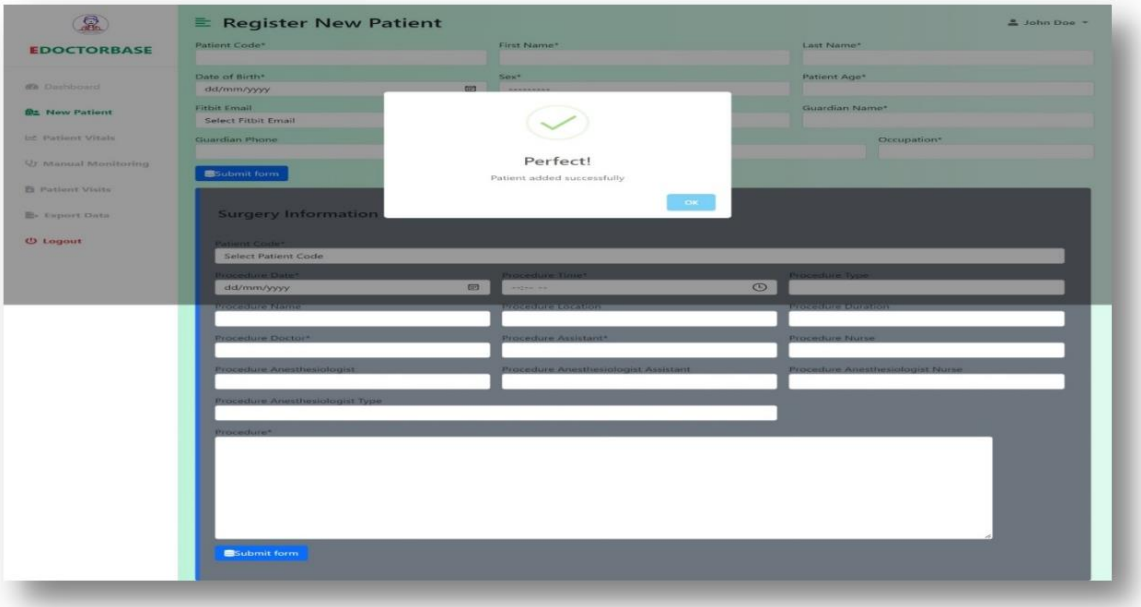


Figure 3.6 Patient Registration

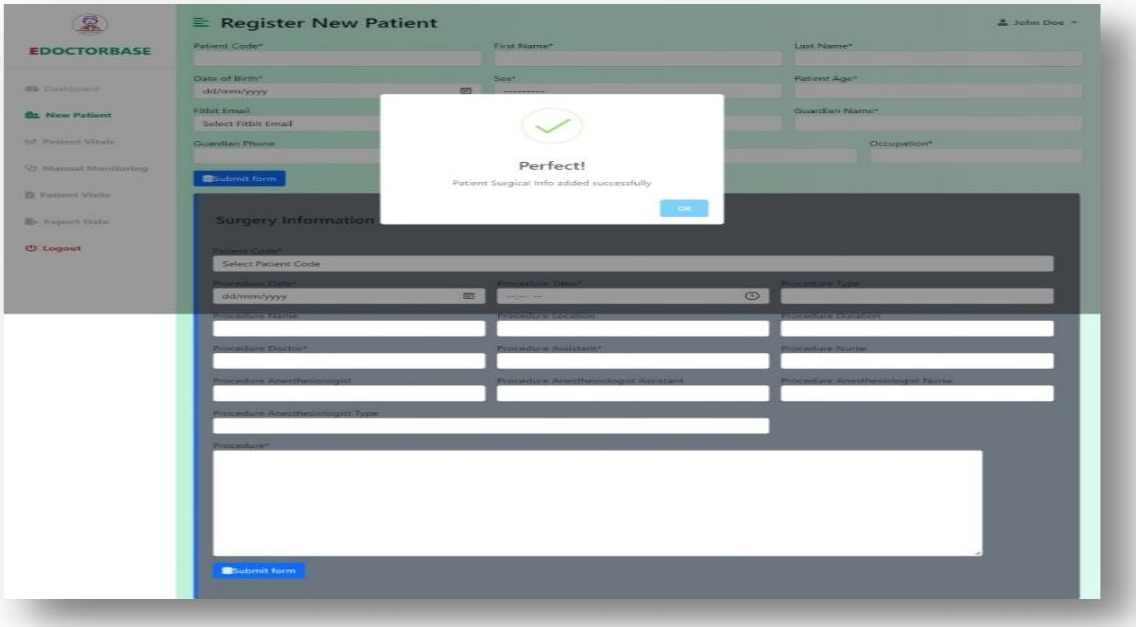


Figure 3.7 Successful upload of patient information

3.7 Upload Patient Vitals Functionality

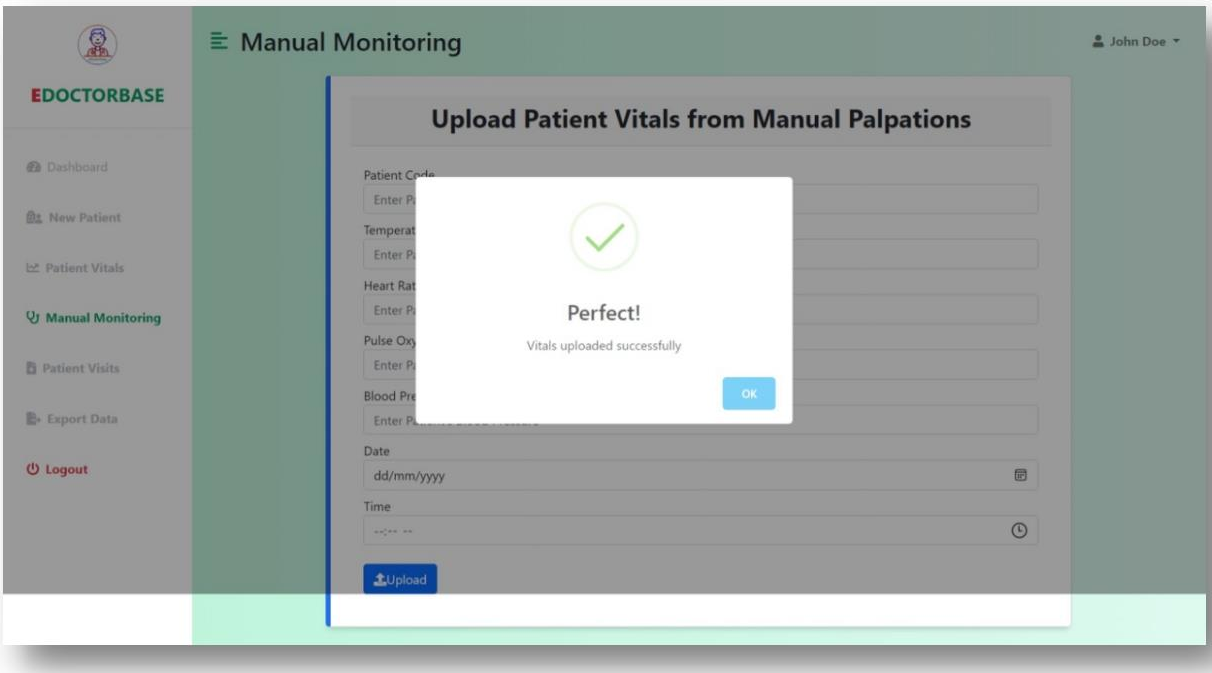


Figure 3.8 Successful upload of patient's vitals

3.8 View Patient Vitals Functionality

Figure 3.9 Patient's Vitals Functionality

3.9 Alert Functionality

Figure 3. 10 Alert Functionality - *shows a table alerting the user for abnormal vitals with color schemes*

3.10 Assigned to Fitbit Email Functionality

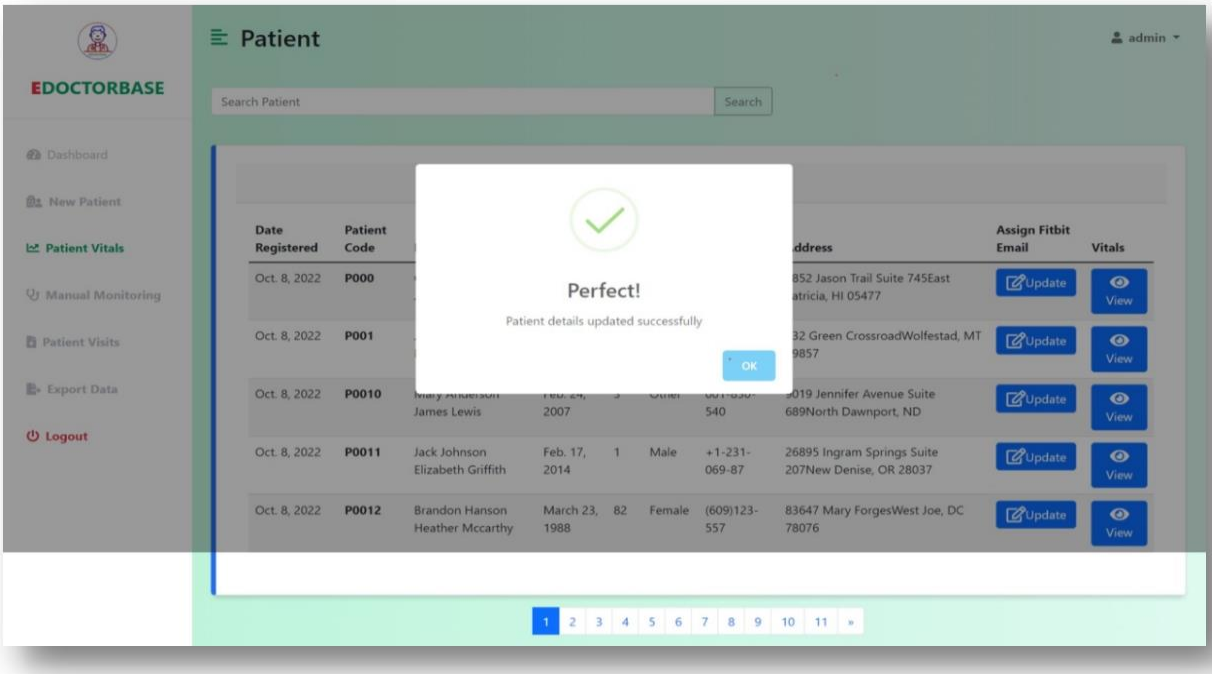


Figure 3.11 Successful assignment of patient to Fitbit email

FITBIT EMAIL	ACCESS TOKEN	USER ID
S		
invalidfitbit@gmail.com	*sfdsffyguihgdg772136*	*1212131*

Table 3.1 shows the list of Fitbit emails with its parameters

3.11 Create Report Functonality

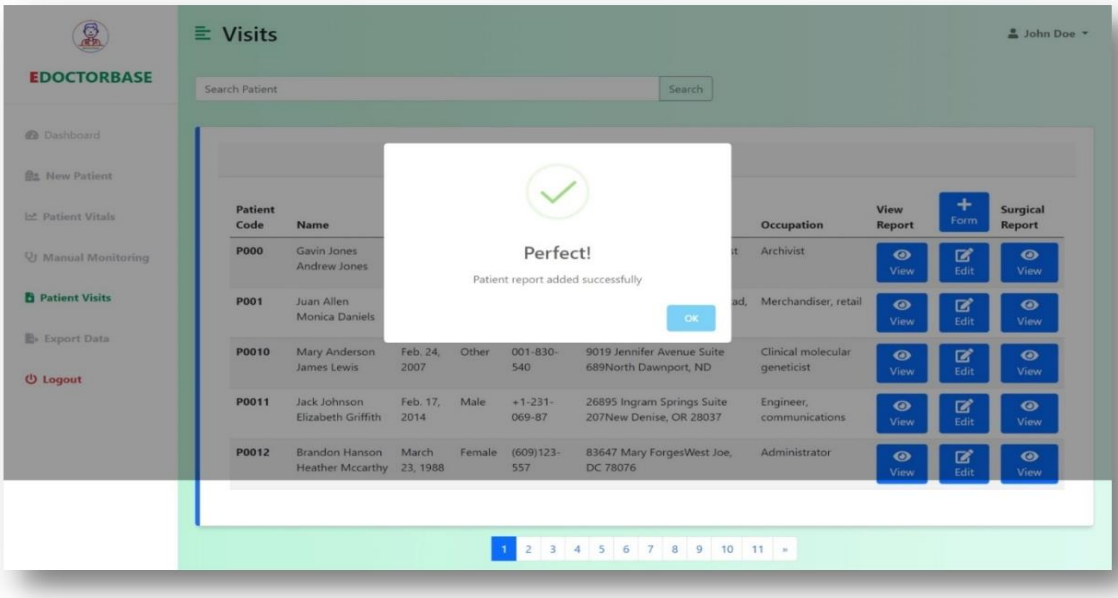


Figure 3.12 Successful upload of patient's vitals

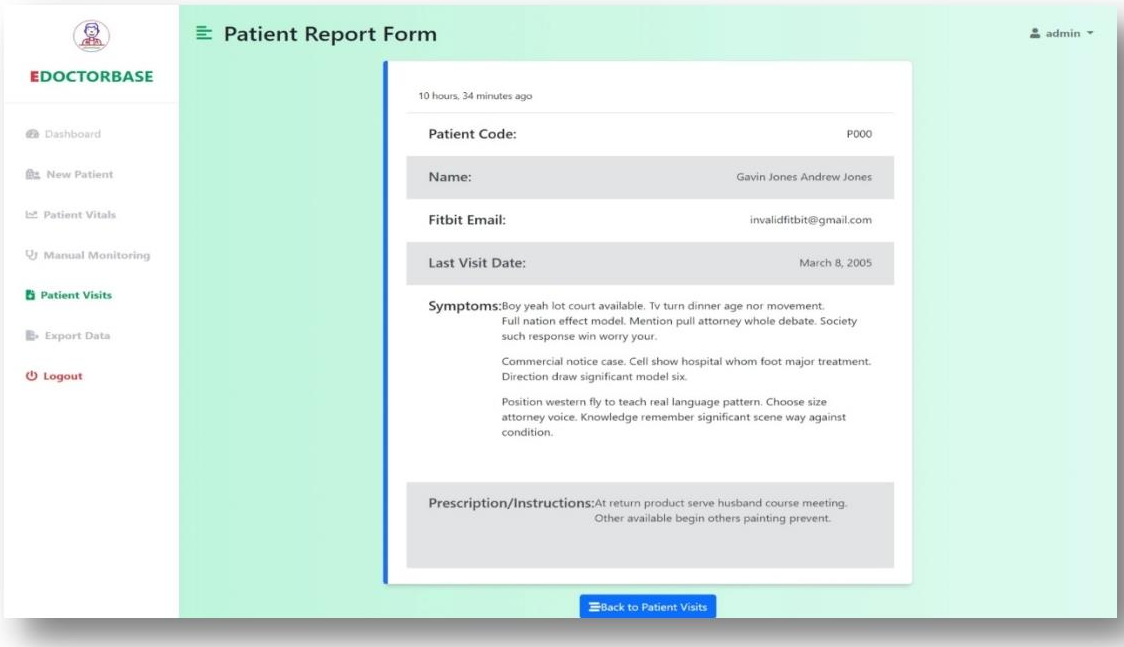


Figure 3.13 Doctor's report

3.12 Export Data

The following figures shows the successful exportation of patient P000 information

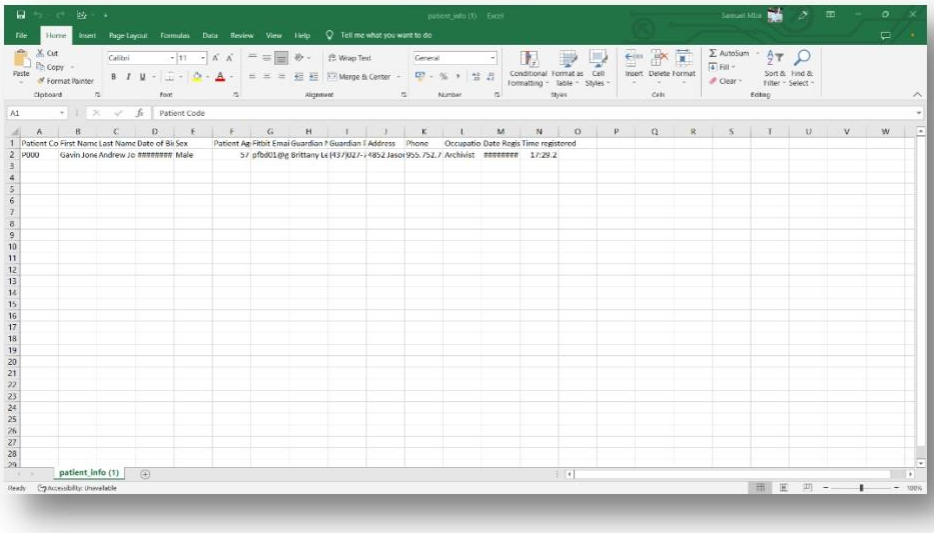


Figure 3.14 Patient's personal information

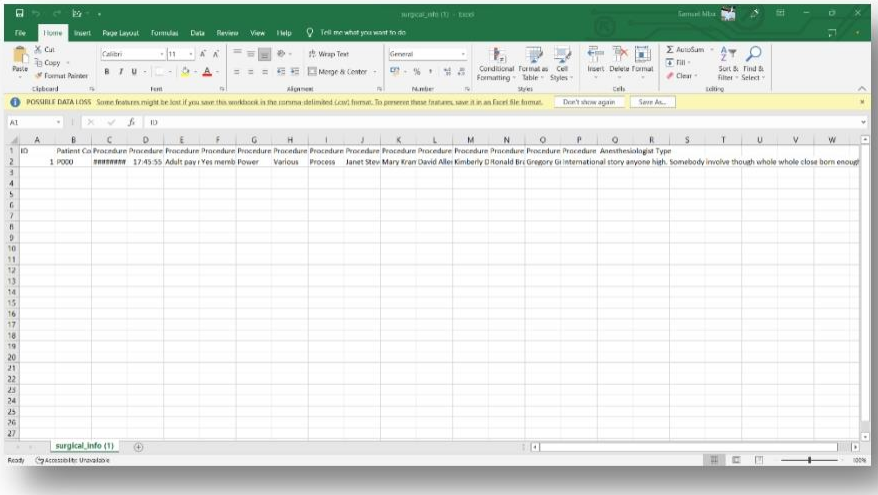


Figure 3.15 Patient's surgical information

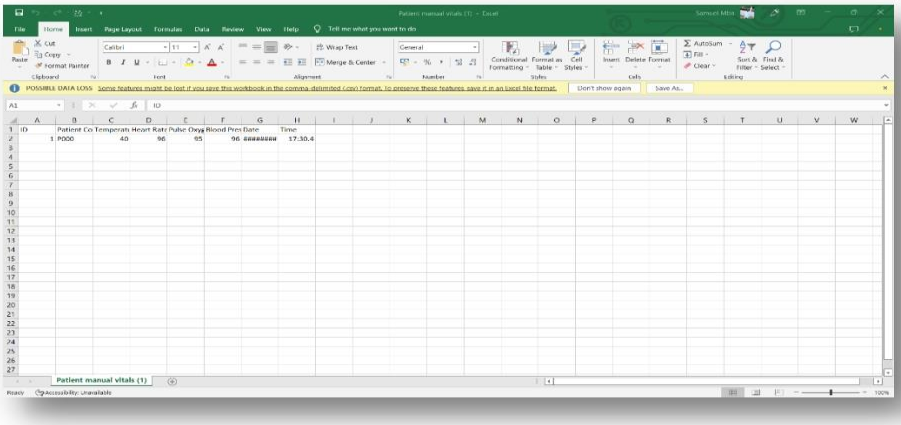


Figure 3.16 Patient's manual vitals

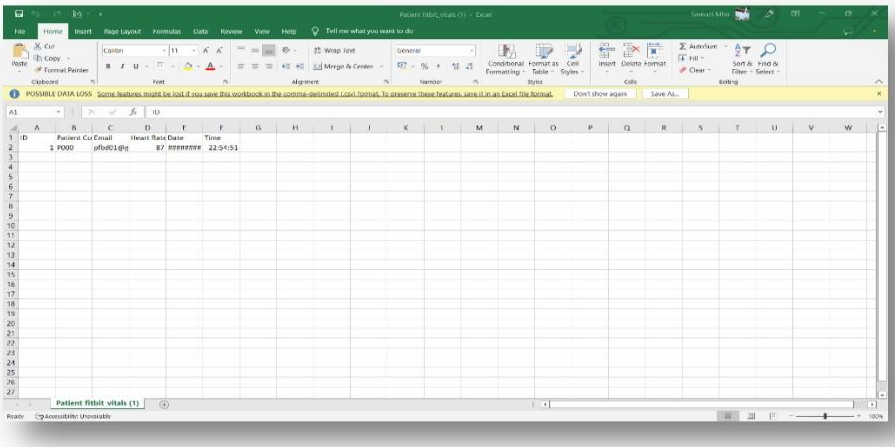


Figure 3.17 Patient's Fitbit vitals

3.13 Database Tables Migration to Heroku's Postgres Server

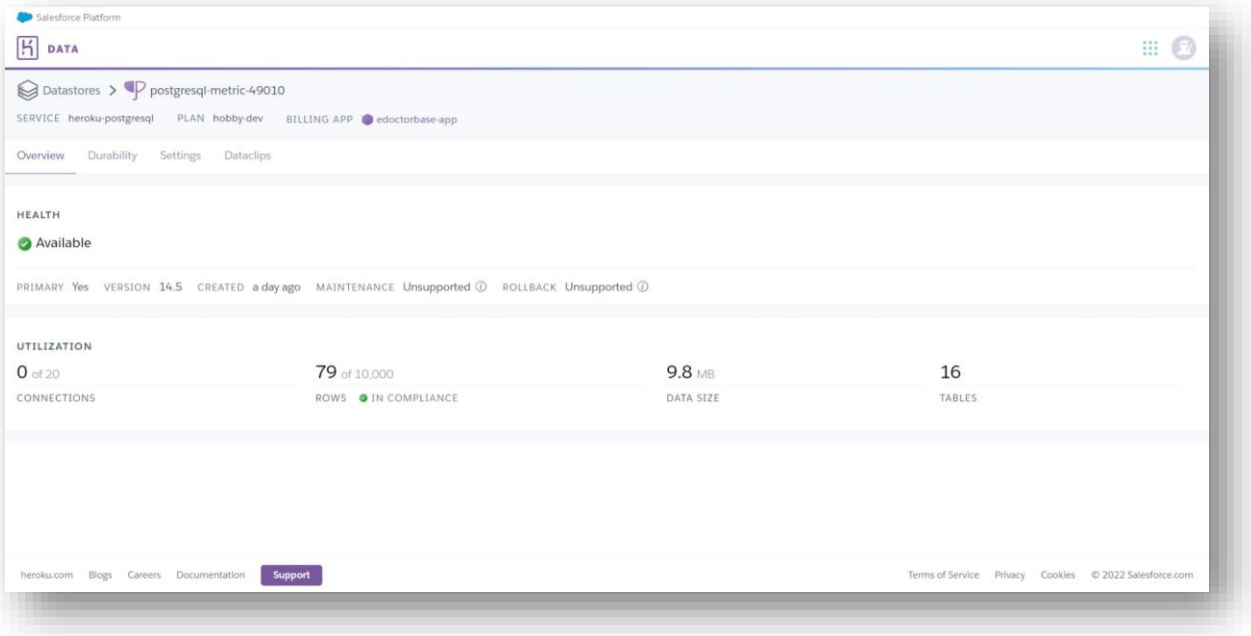


Figure 3.18 Migration of the database tables to postgres DB

Chapter4 Data Collection, Processing and Statistical Analysis

4.1 Participants

Due to unavailability of approval from the FDA at the time of the study, the team had to compromise by using a volunteer who was also part of the project team. The volunteer is a healthy 22-year-old male with no history of any medical condition. The original study was to be conducted on 30 patients but due to the unavailability of the wearable devices to the team at the time of this study, the study was limited to only one volunteer.

4.2 Devices and gold standard

One of the study objectives was to investigate the accuracy of Xiaomi Mi Band 6 and Fitbit inspire 2 and in the long run use them as alternatives and/or replacement for manual mode of vital sign measurement in various hospitals in Ghana. The Xiaomi Mi Band 6 smart wearable device was released on the 29th of March 2021. It is able to record heart rate, SpO2 and also have a feature for sleep monitoring. The Mi band 6 has a battery life of 14 days, which makes it very suitable for the study since the volunteer was going to wear it for long hours. The Fitbit Inspire 2 was not used for the study due to its unavailability at the time of this study so Fitbit Versa 2, which was readily available, was rather used for the study. Fitbit Versa 2 was released by the Fitbit Company on the September 15 2019. It is able to record heart rate data as well as track one's sleep. Fitbit uses sensors made up of photodiodes to measure heart rate of an individual by a method called photoplethysmography [1]. When an individual's heartbeat, the capillaries expand and contract based on the changes in blood volume. In determining the heart rate, the photodiodes from the Fitbit flashes light on the capillaries to determine the changes in volume of blood above the wrist. The number of times the heart beats per minute is being calculated. It has a battery life of six or more days based on manufacturer specifications and is suitable for this study for the same reason stated for the Mi band 6. Two devices were considered used as the gold standard of vital sign measurement for the study. They are the PHILIPS MX 450 patient monitor and a portable Blood pressure monitoring device (Balance brand). The PHILIPS MX 450 patient monitor uses a five lead ECG system for heart rate measurement and a pulse oximeter connected to it for SpO2 measurement. The portable manual monitor is made of a blood pressure cuff that determines the blood pressure as well as heart

rate. The two wearable devices (Fitbit Versa 2 and Xiaomi Mi band 6) were worn on the same arm on the volunteer and the 5 ECG leads were placed as shown in Figure 46 and 47 below.

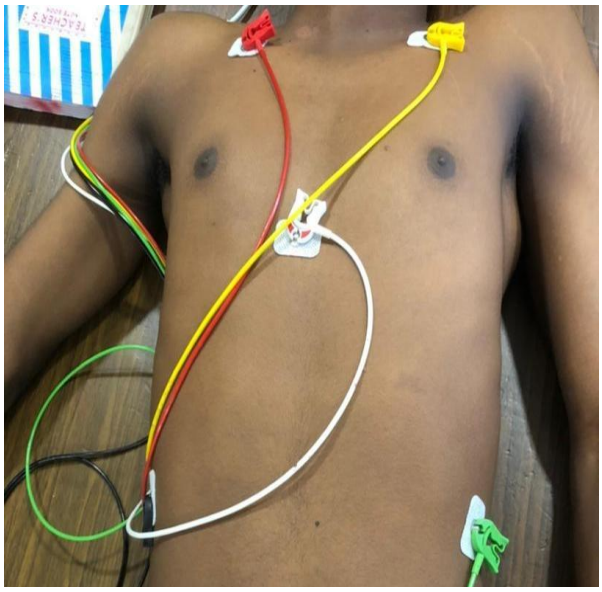


Figure 4.1 5 lead ECG placement

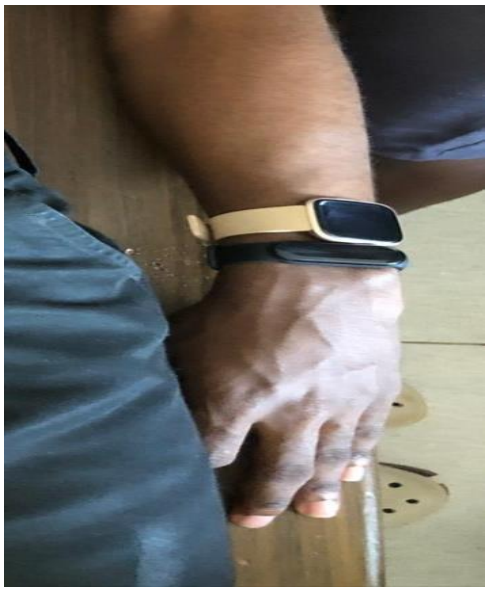


Figure 4.2 Wearable devices worn on volunteer

4.3 Study Procedure

The study was carried out in a closed room environment at the University of Ghana Medical Center (UGMC) on the 18th and 19th of August 2022. The study was conducted under a normal environment with temperature around 25 degrees Celsius. The two devices were placed on the dominant arm and synchronized with their respective mobile application on a smart phone. There was not much noise produced in heart rate measurement from the ECG leads since the volunteer did not have much hair on the body. Recording of vital signs commenced at least one minutes after placement of the ECG electrodes. The volunteer lied on a table in a supine position to ensure as previously done in a similar study [35]. The heart rate and SpO2 data from the PHILIPS MX 450 patient monitor were recorded in intervals of one minute and that of the wearable devices were also recorded in intervals of one minute as well.

4.4 Data Collection and Processing

The PHILIPS MX 450 patient monitor was set to record heart rate and SpO2 data per every minute. One member of the team had the task of recording the heart rate and SpO2 data from the PHILIPS MX 450 patient monitor into a CSV file. The data came with the time at which it was recorded and this was important because data from each device were matched based on

time stamps. A total of 116 heart rate data points were recorded by the patient monitor on day 1 and 121 heart rate data points on day 2 summing up to a total of 237 heart rate data points obtained from the PHILIPS MX 450 patient monitor. The PHILIPS MX 450 patient monitor also recorded 116 SpO2 data points on day 1 and 121 on day 2 also summing up to a total of 237 SpO2 data points.

In order to retrieve data recorded by Fitbit, an API was registered on the Fitbit website and then a python file was written to make a get request to the registered API with Fitbit. The data was retrieved in a CSV file format with time stamps and in intervals of one minute. The retrieved CSV file from Fitbit contained 135 heart rate data points for day 1 and 121 for day 2 summing up to a total of 256 heart rate data points.

To retrieve data from Xiaomi, an account was registered with a third-party website called huami. Data recorded on the Xiaomi device was retrieved in a CSV file format via the huami website. The data recorded by the Xiaomi device is made of 130 heart rate data points on day 1, 121 on day 2 making a total of 251 heart rate data points. Data retrieved from Xiaomi device was also recorded every minute and came with time of recording as well. The Xiaomi device also recorded SpO2 with 36 data points on day 1 and 115 data points on day 2 and an overall of 151 SpO2 data points. The SpO2 data was recorded manually and processed into a CSV file. Heart rate data obtained from all the devices were processed into a single CSV file for each day and then combined to form an overall heart rate data for each device in a single CSV file making a total of three CSV files for day 1, day 2 and for overall. The data from all devices were matched using the time at which they were recorded. A total of 116 data points were processed into a single CSV file for each device on day 1, 121 heart rate data points for day 2 and an overall of 237 data points for heart rate. 36 SpO2 data points were processed into a CSV file for PHILIPS MX 450 patient monitor and Xiaomi on day 1, 115 SpO2 data points on day 2 and an overall of 151 data points for SpO2.

To obtain manual data from the portable Blood Pressure monitor, the BP cuff was worn on the left arm around the biceps and triceps 2-3 cm from the elbow and fastened tightly. The cuff was connected to the BP device, which provided pressure on the arm for the physiological data to be measured. Heart rate data was recorded in intervals of 30 minutes and then later in intervals of 1 minute, (this was done to obtain a sufficient number of data points). The time at which the data was taken was then recorded. A total of 116 heart rate data points was obtained.

4.4.1 Summary of Data Collected

DAY 1

Device / Vital Sign	PHILIPS MX 450 patient monitor	Xiaomi Mi Band 6	Fitbit Versa 2
Heart Rate	116	130	135
Oxygen Saturation (SpO2)	116	36	-

Table 4.4.1 shows a summary of data collected for Day 1

DAY 2

Device / Vital Sign	PHILIPS MX 450 patient monitor	Xiaomi Mi Band 6	Fitbit Versa 2
Heart Rate	121	121	121
Oxygen Saturation (SpO2)	121	115	-

Table 4.4.2 shows a summary of data collected for Day 2

4.5 Statistical Analysis

All statistical analysis on data collected and processed were done in Python (Jupyter notebook) which was locally installed on a computer. The python libraries used were numpy, scipy, matplotlib, seaborn, stats models, and pandas. In order to use the right statistical measure, normality of data collected was tested to identify whether or not the dataset follows a Gaussian distribution in order to select the right tests for the data set [36]. To achieve this, a quantile-quantile (Q-Q) plot, histogram plot and Shapiro-Wilk test were used. All test proved that the data set does not follow a normal distribution and therefore only statistical analysis valid for non-normally distributed dataset were used to evaluate the performance of the wearable devices (Fitbit and Xiaomi) against the patient monitor and the manual monitoring device.

4.6 Normality testing

Figure 4.3 and 4.4 show the normality test for patient monitor heart rate dataset.

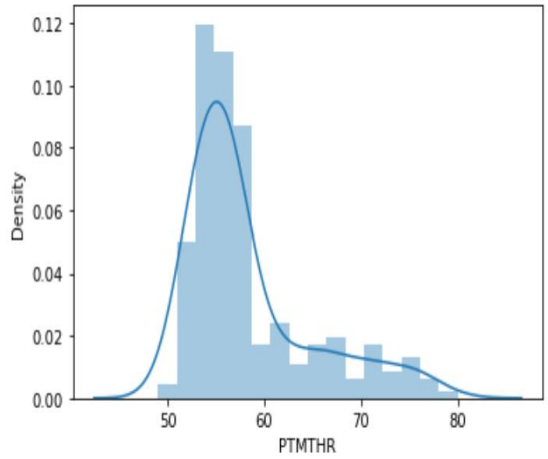


Figure 4.3 Patient monitor histogram plot for HR.

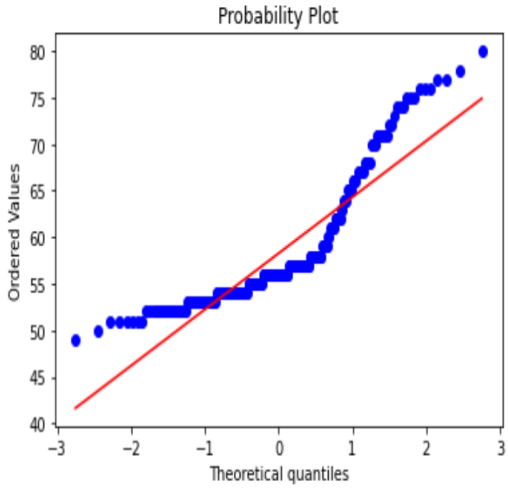


Figure 4.4 Patient monitor Q-Q plot for HR.

Figure 4.5 and Figure 4.6 show the normality test for Xiaomi heart rate dataset whilst Figure 4.7 and Figure 4.8 shows that of Fitbit heart rate dataset.

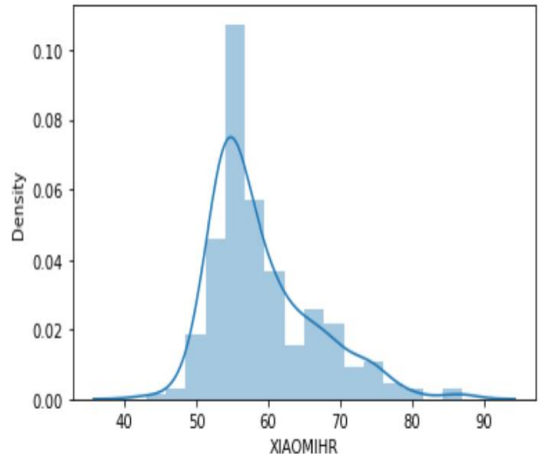


Figure 4.5: Xiaomi HR histogram plot

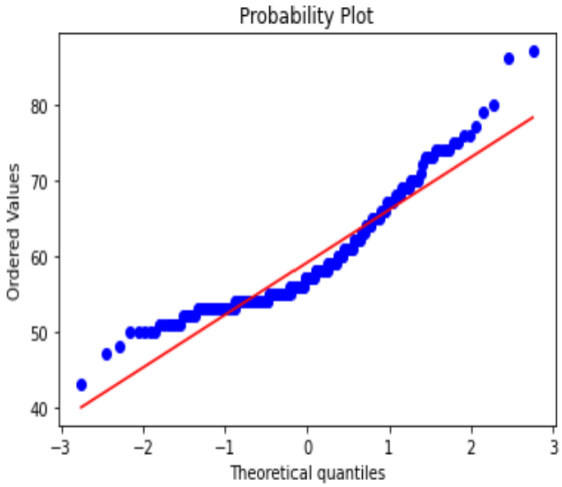


Figure 4.6**:** Xiaomi HR Q-Q plot

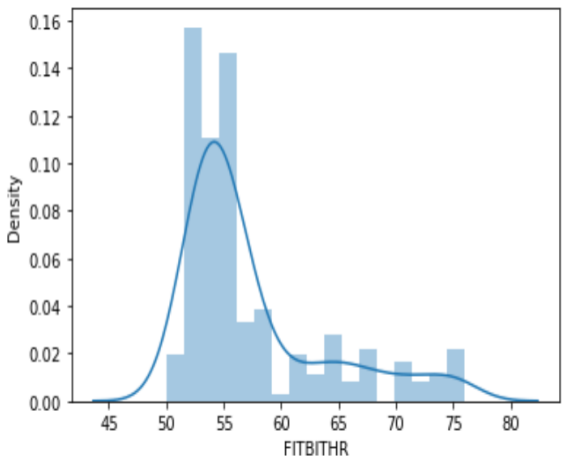


Figure 4.7: Fitbit HR Histogram plot

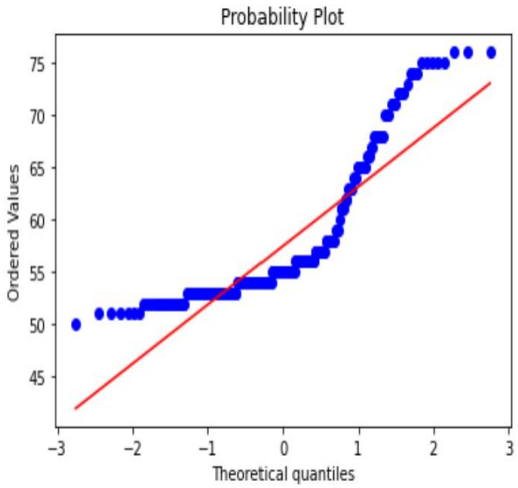


Figure 4.8: Fitbit HR Q-Q plot

Figure 4.9 and Figure 4.10 shows the normality test for PHILIPS MX 450 monitor SPO2 dataset whilst Figure 4.11 and Figure 4.12 shows SpO2 for Xiaomi

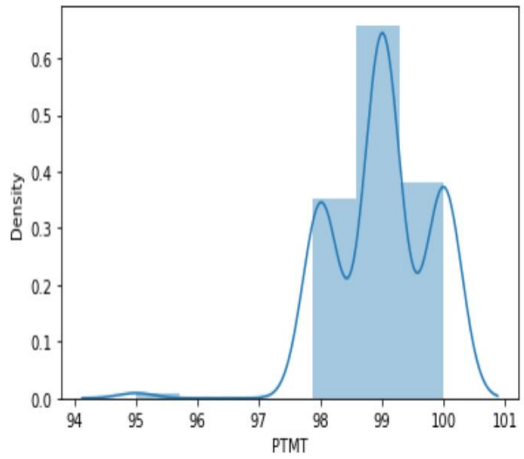


Figure 4.9: Patient monitor spo2 histogram plot

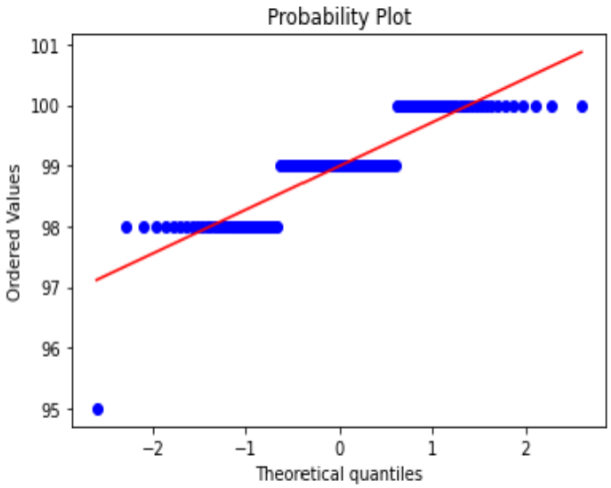


Figure 4.10: Patient monitor spo2 Q-Q plot

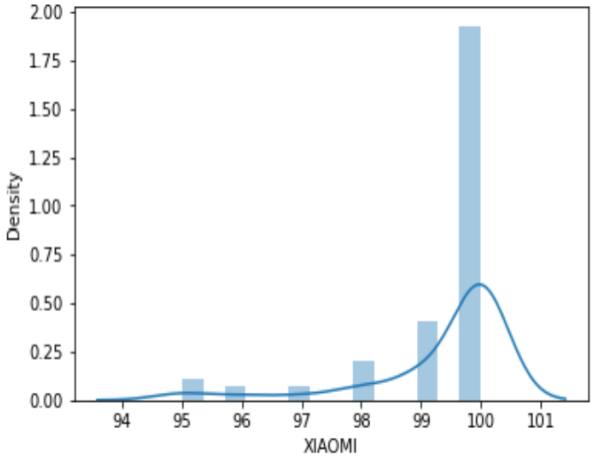


Figure 4.11: Xiaomi spo2 histogram plot

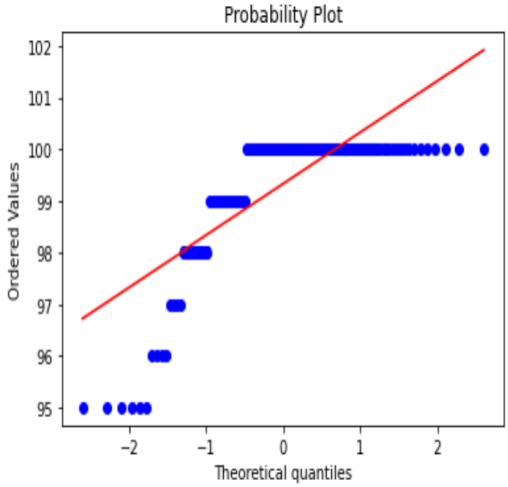


Figure 4.12: Xiaomi spo2 Q-Q plot

Figure 4.13 and Figure 4.18 shows the normality test fot heart rate data when the manual monitor was used. This test includes that of Fitbit, Xiaomi and the manual monitoring method.

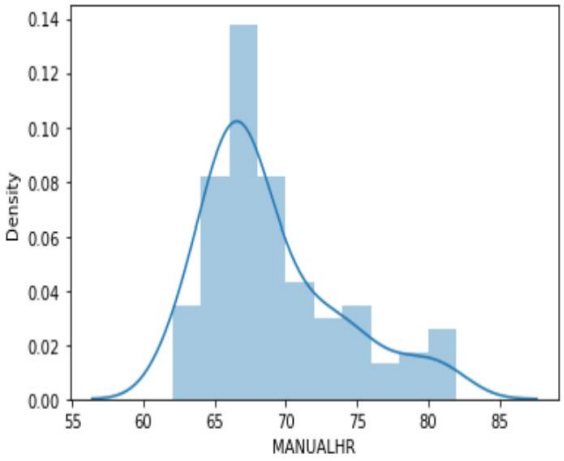


Figure 4.13: Manual monitor HR histogram plot

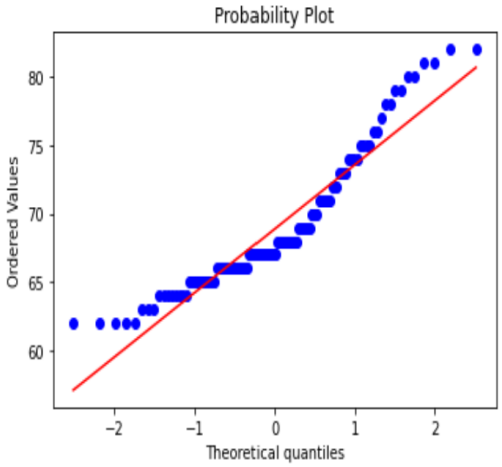


Figure 4.14: Manual monitor HR Q-Q plot

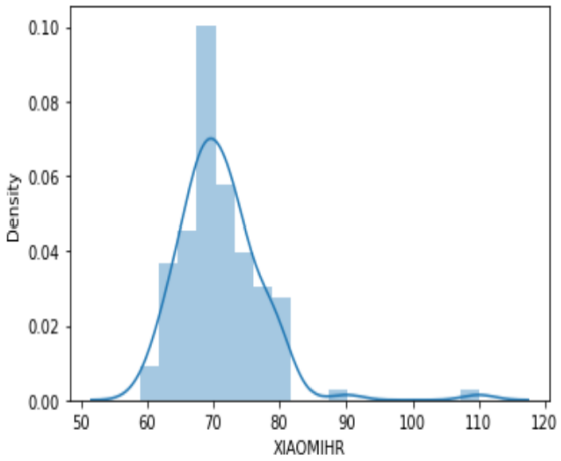


Figure 4.15: Xiaomi HR histogram plot

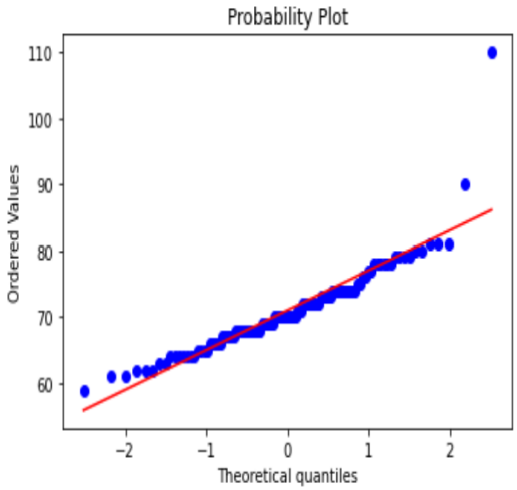
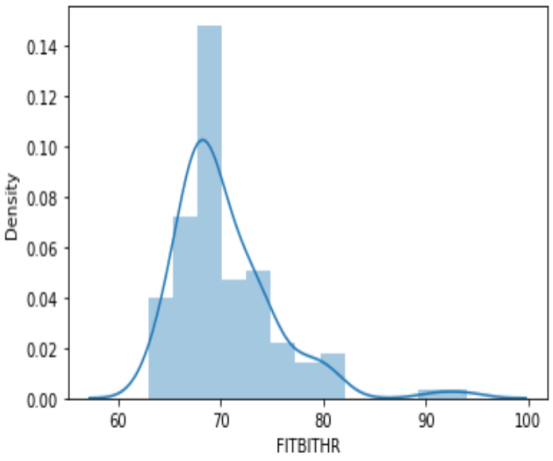


Figure 4.16: Xiaomi HR Q-Q plot



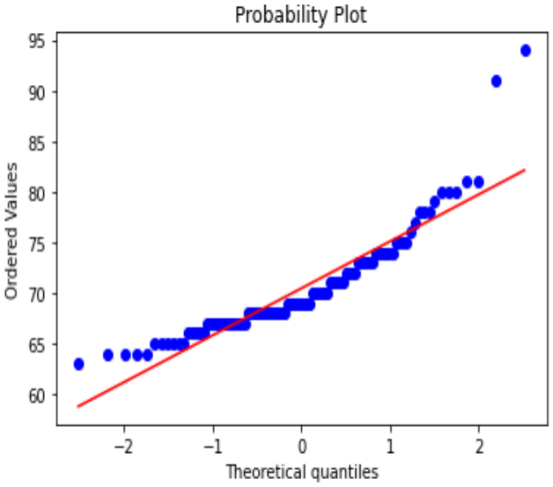


Figure 4.17: Fitbit HR histogram plot

Figure 4.18: Fitbit HR Q-Q plot

The table below shows the P values gotten for the Shapiro-Wilk normality test.

Vital Sign	P value	Conclusion
Patient monitor heart rate	1.06×10^{-15}	Not normally distributed
Xiaomi heart rate on same day with patient monitor	1.60×10^{-11}	Not normally distributed
Fitbit heart rate on same day with patient monitor	1.34×10^{-17}	Not normally distributed
Xiaomi SPO2 data	8.31×10^{-19}	Not normally distributed
Patient monitor SPO2 data	6.60×10^{-13}	Not normally distributed
Manual heart rate data	4.22×10^{-7}	Not normally distributed
Xiaomi heart rate on same day with manual monitor	2.38×10^{-9}	Not normally distributed
Fitbit heart rate on same day with manual monitor	6.06×10^{-9}	Not normally distributed

Table 4.6.1 shows the summary of P-values for Shapiro Willk test

Three major analysis were performed on the datasets in python. The first is correlation analysis. The pearson correlation coefficient was calculated in python using the in-built libraries. The pearson correlation coefficient provides a measure on the relationship between two variables and in this case two different methods of measurement [37]. It is very important to know whether or not Fitbit and Xiaomi will measure any decrease or increase recorded by the gold standard devices (PHILIPS MX 450 patient monitor and manual vital monitoring device). The Bland-Altman analysis was also performed between the various devices. The Bland-Altman analysis is a plot that shows the level of agreement between a new method of measurement and an already existing method of measurement by estimating the mean difference and the level of agreement between the two methods used in measuring the same quantity over 95% confidence interval [38].The plot also gives information on whether or not the new method is over estimating or under estimating the measurement as well as possible outliers. The description given above for the Bland-Altman plot makes it suitable for the study. The final metric which was evaluated was the mean absolute percentage error (MAPE). The MAPE gives errors in estimating a quantity by a new method of measurement relative to a gold standard method in

percentage [39]. MAPE is very important as it was used to determine the accuracy of Fitbit and Xiaomi devices.

4.7 Results

4.7.1 Pearson correlation coefficient

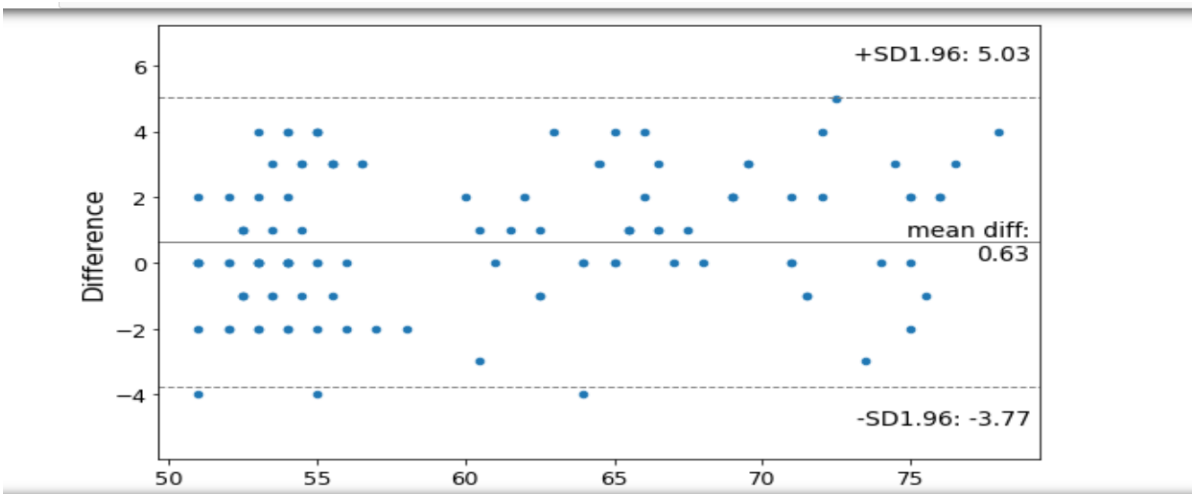
The table below shows the pearson correlation coefficient that were gotten during the analysis in python.

	PHILIPS MX 450 patient monitor heart rate	Manual Monitor Heart rate
Fitbit Heart rate	0.935949	0.825244
Xiaomi Heart rate	0.749383	0.422386

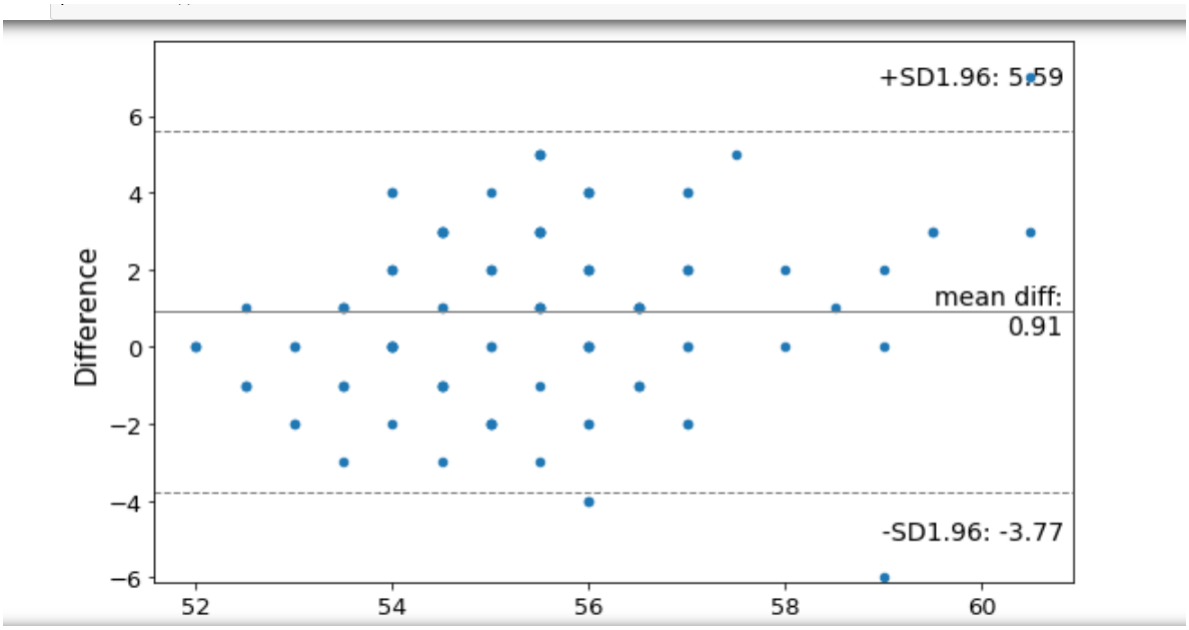
Table 4.7.1 Pearson Coefficient - shows values of Pearson Correlation Coefficient for Fitbit and Xiaomi Devices against the Gold Standard

4.7.2 Bland-Altman Analysis

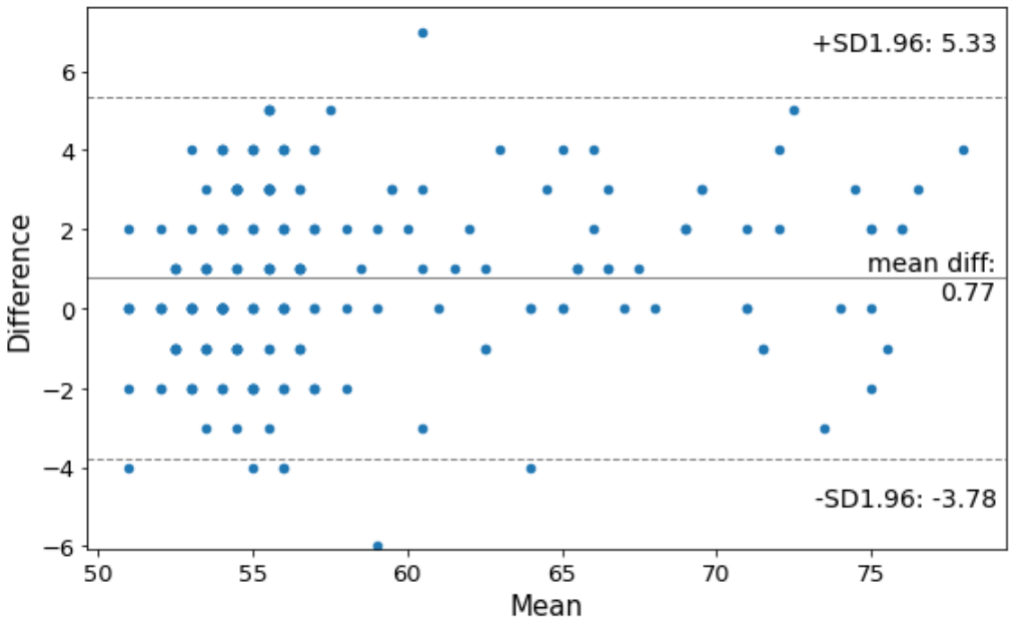
The following results were obtained from the Bland-Altman plot. For Fitbit heart rate against the PHILIPS MX 450 patient monitor heart rate on day 1 of the study, the plot showed a mean difference of 0.63 and limits of agreement (-3.77 to 5.03) with 3 outliers as shown below.



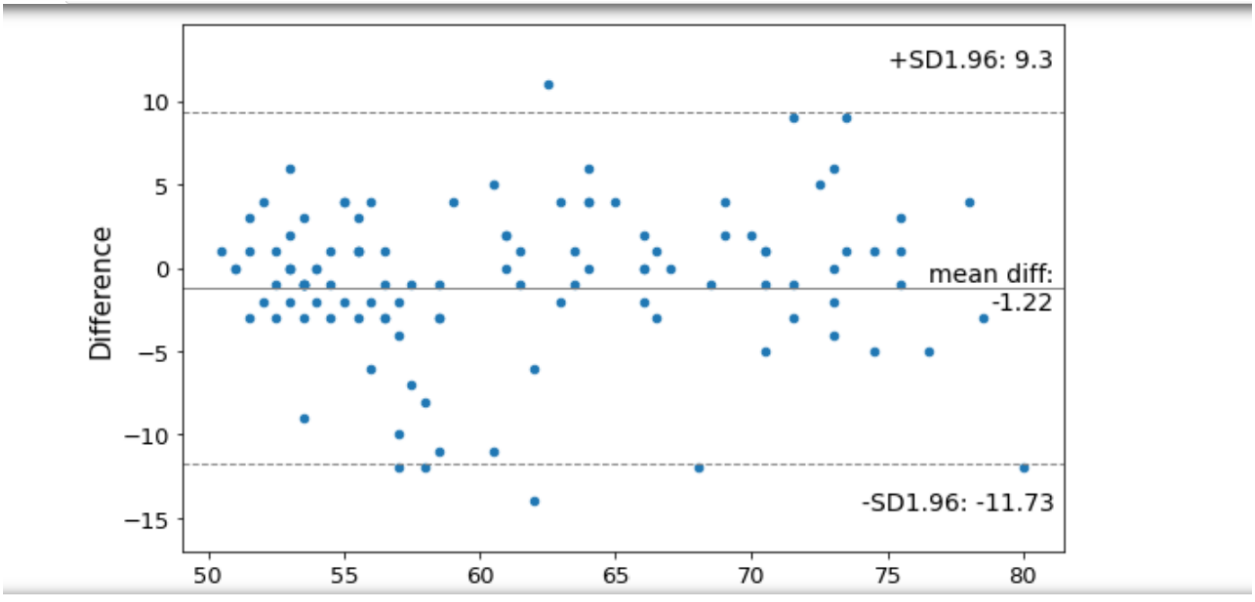
The Bland-Altman plot for Fitbit heart rate against the PHILIPS MX 450 patient monitor heart rate on day 2 of the study, the plot showed a mean difference of 0.91 and limits of agreement (-3.77 to 5.59) with 3 outliers as shown below.



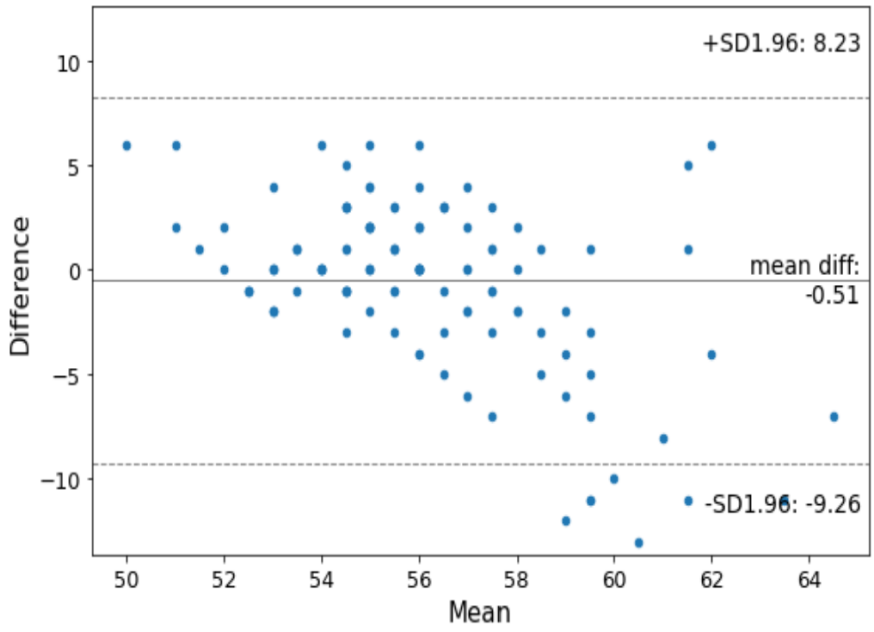
The Bland-Altman plot for the overall Fitbit heart rate against the PHILIPS MX 450 patient monitor heart rate , the plot showed a mean difference of 0.77 and limits of agreement (-3.78 to 5.33) with 6 outliers as shown below.



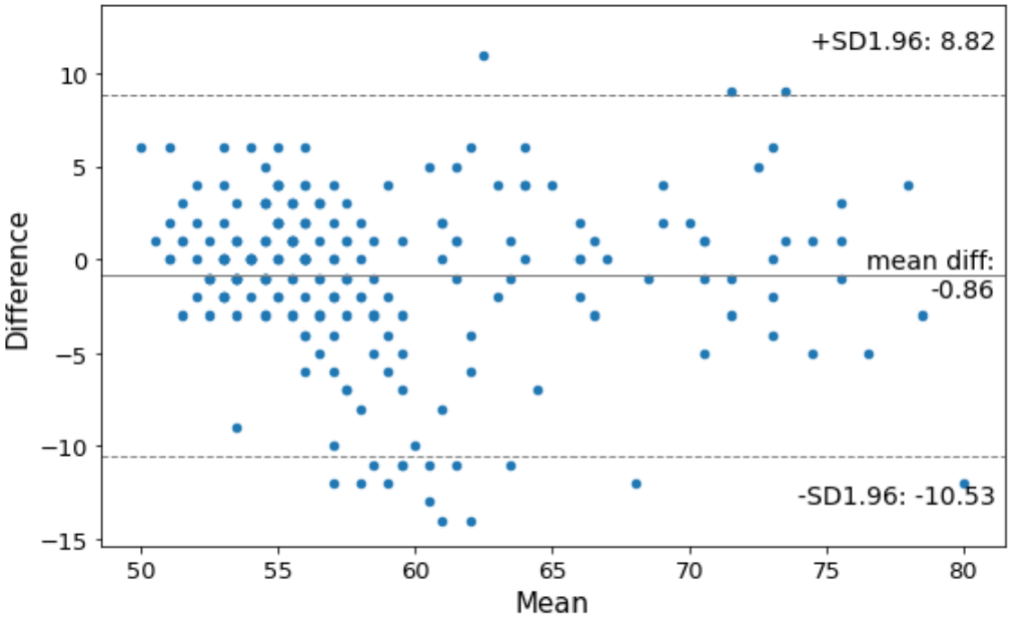
The Bland-Altman plot for Xiaomi heart rate against the PHILIPS MX 450 patient monitor heart rate on day 1 of the study, the plot showed a mean difference of -1.22 and limits of agreement (-11.73 to 9.3) with 6 outliers as shown below.



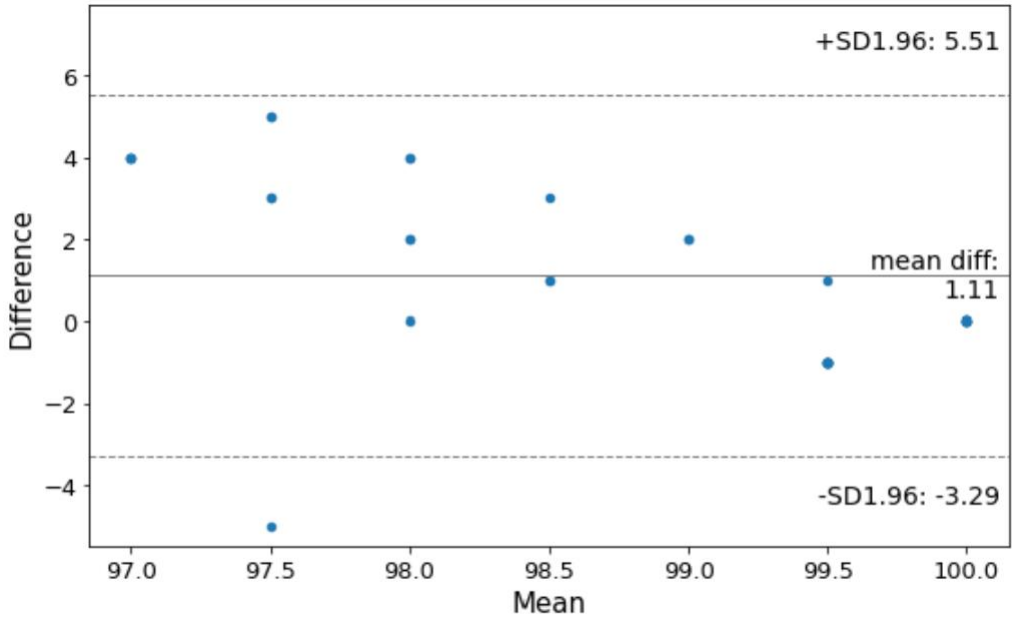
The Bland-Altman plot for Xiaomi heart rate against the PHILIPS MX 450 patient monitor heart rate on day 2 of the study, the plot showed a mean difference of -0.51 and limits of agreement (-9.26 to 8.23) with 6 outliers as shown below.



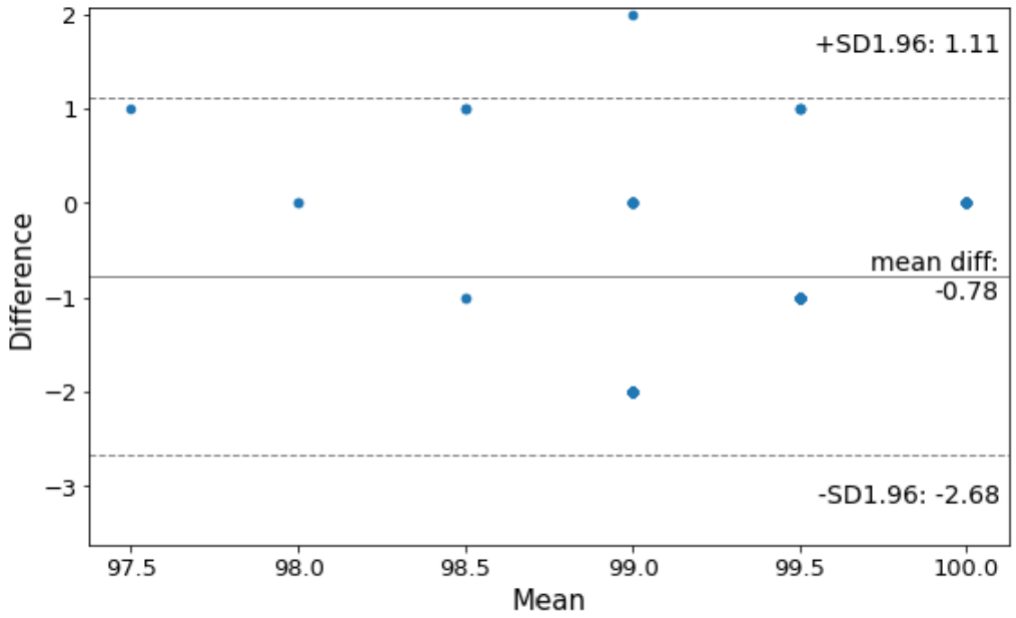
The Bland-Altman plot for the overall Xiaomi heart rate against the PHILIPS MX 450 patient monitor heart rate, the plot showed a mean difference of -0.88 and limits of agreement (-10.53 to 8.82) with 15 outliers as shown below.



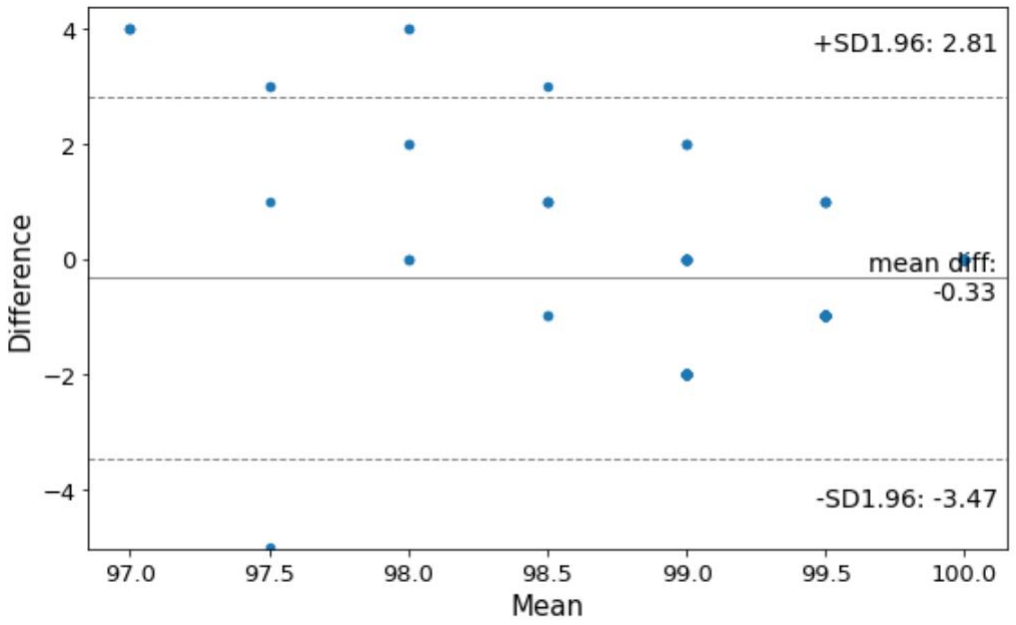
The Bland-Altman plot for Xiaomi SpO2 against the PHILIPS MX 450 patient monitor SpO2 on day 1 of the study, the plot showed a mean difference of 1.11 and limits of agreement (-3.29 to 5.51) with no outliers as shown below.



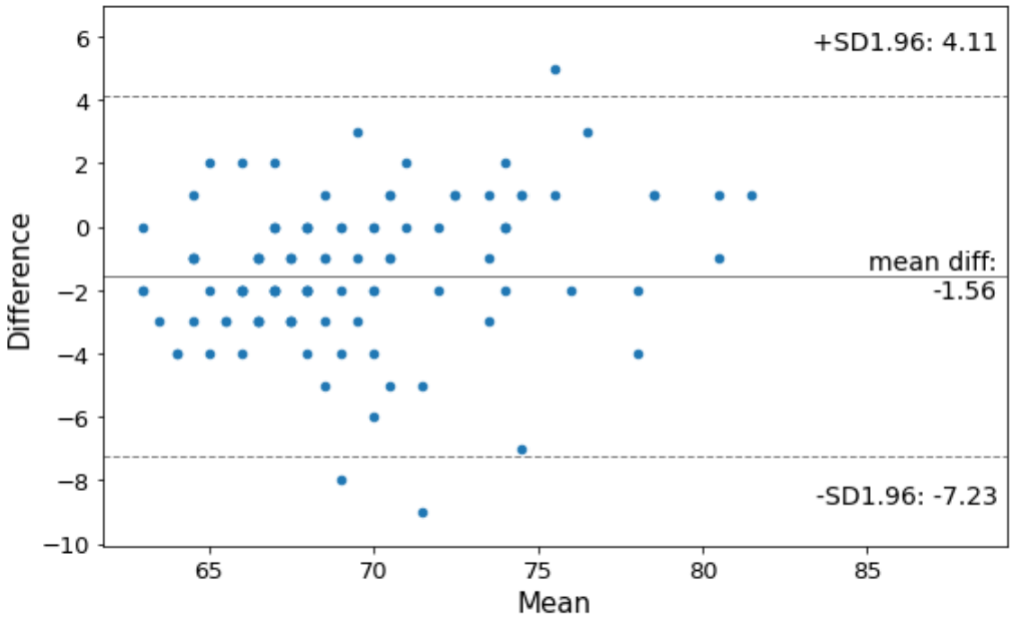
The Bland-Altman plot for Xiaomi SpO2 against the PHILIPS MX 450 patient monitor SpO2 on day 2 of the study, the plot showed a mean difference of -0.78 and limits of agreement (-2.68 to 1.11) with 1 outlier as shown below.



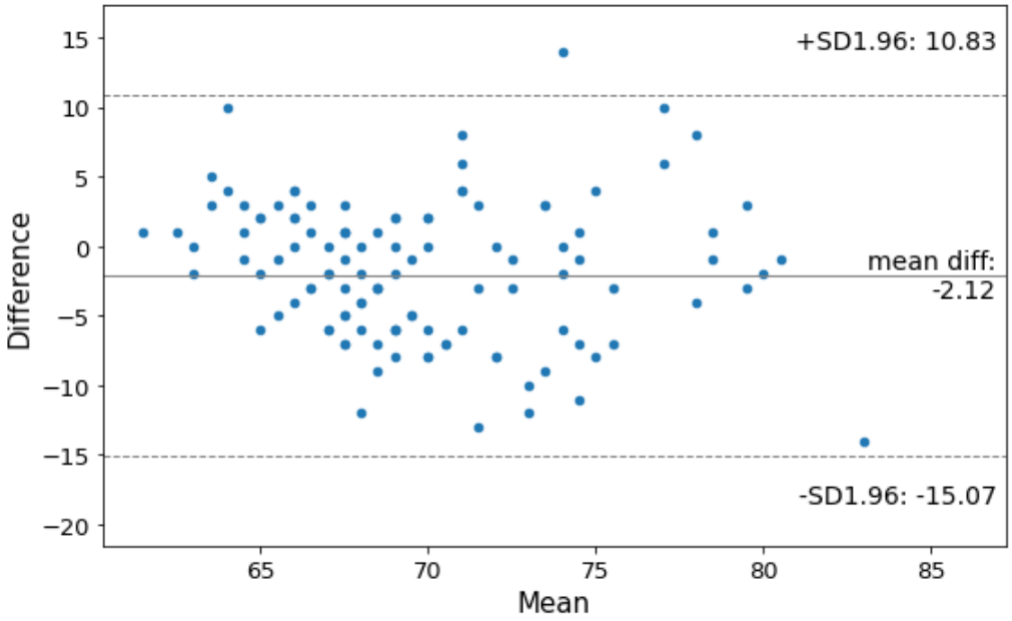
The Bland-Altman plot for the overall Xiaomi SpO2 against the PHILIPS MX 450 patient monitor SpO2, the plot showed a mean difference of -0.33 and limits of agreement (-3.47 to 2.81) with 5 outliers as shown below.



The Bland-Altman plot for Fitbit heart rate data against the manual vital sign monitor, the plot showed a mean difference of -1.56 and limits of agreement (-7.23 to 4.11) with 3 outliers as shown below.



The Bland-Altman plot for Xiaomi heart rate data against the manual vital sign monitor, the plot showed a mean difference of -2.12 and limits of agreement (-15.07 to 10.83) with 1 outlier as shown below



4.7.3 Mean Absolute Percentage Error

The table below shows results obtained from the various vital sign datasets for mean absolute percentage error (MAPE)

	Fitbit heart rate day 1	Fitbit heart rate day 2	Overall Fitbit heart rate	Xiaomi heart rate day1	Xiaomi heart rate day2	Overall Xiaomi heart rate	Xiaomi SPO2 day 1	Xiaomi SPO2 day 2	Overall Xiaomi SPO2
Patient monitor	2.84	3.4	3.13	6.15	5.39	5.76	1.92	0.99	1.21
Manual monitor			3.36			6.79			

Table 4.7.3 shows a summary of MAPE Values

4.7.4 Data Visualization

Figure 4.19 and Figure 4.20 show the data visualization of the devices using the line plot made in python.

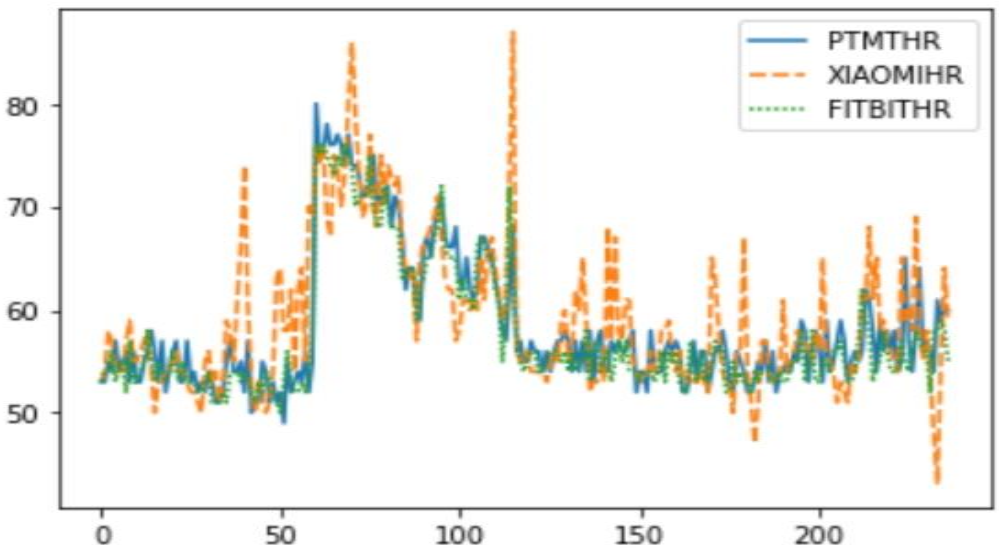


Figure 4.19. Line plot for patient monitor against Fitbit and Xiaomi. ***Key: PTMTHR*** *– Patient Monitor heart rate* ***XIAOMIHR –*** *Xiaomi heart rate.* ***FITBITHR-*** *Fitbit heart rate*

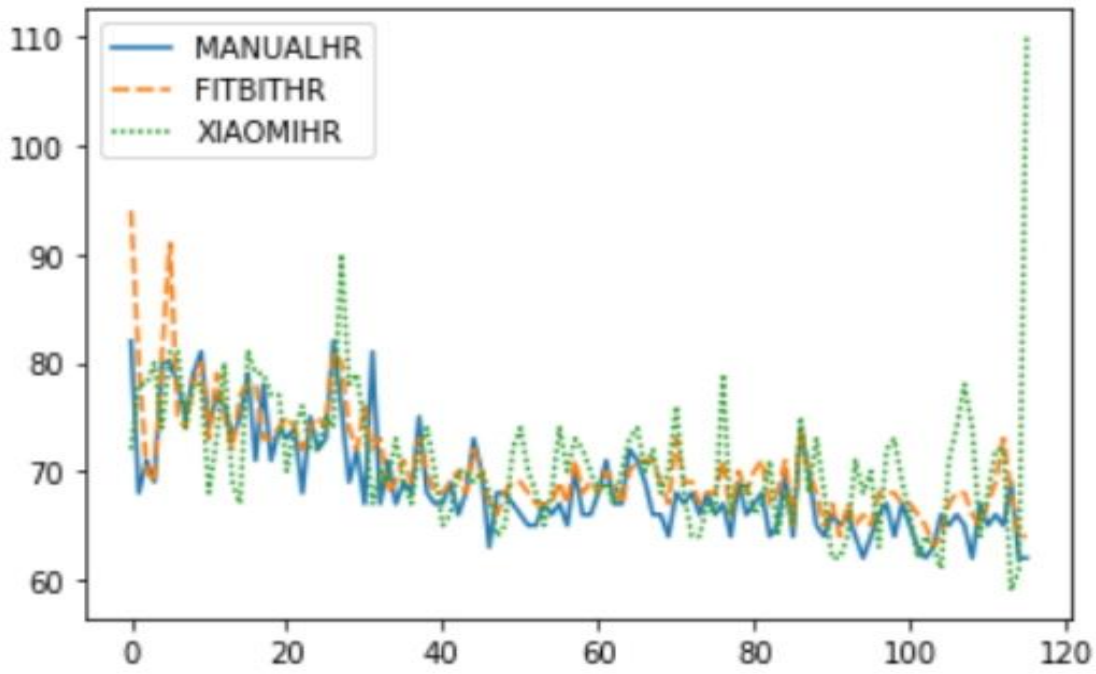


Figure 4.20. Line plot for manual monitor against Fitbit and Xiaomi for heart rate. ***Key: MANUALHR*** *– Patient Monitor heart rate* ***XIAOMIHR –*** *Xiaomi heart rate.* ***FITBITHR-*** *Fitbit heart rate*

Figure 4.21 and Figure 4.22 below shows results of the scatter plot generated for the various wearable devices in comparison to the gold standard.

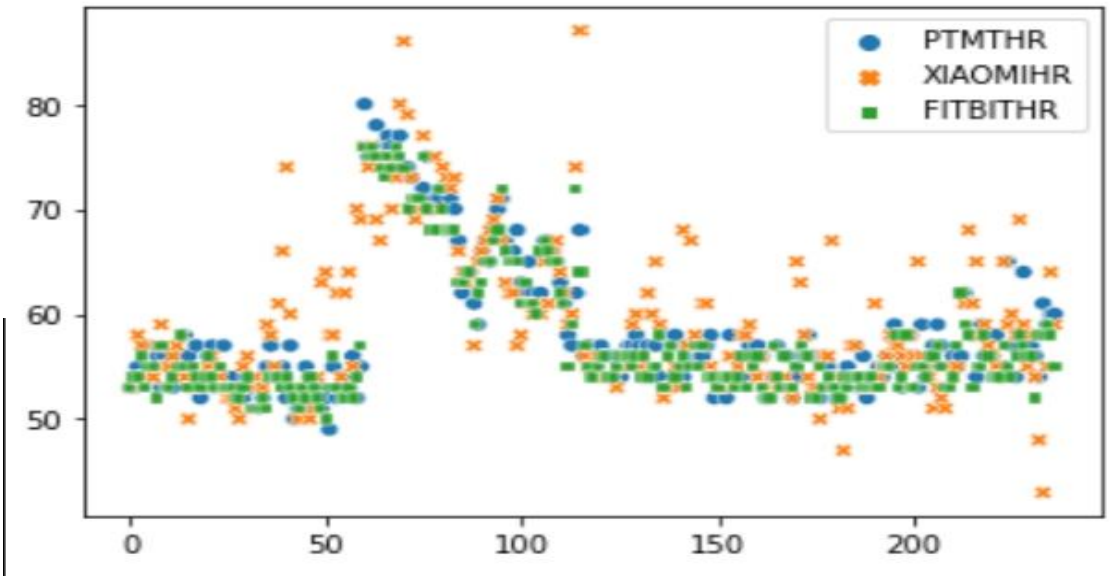


Figure 4.21. Scatter diagram for patient monitor against Fitbit and Xiaomi. ***Key: PTMTHR*** *– Patient Monitor heart rate* ***XIAOMIHR –*** *Xiaomi heart rate.* ***FITBITHR-*** *Fitbit heart rate*

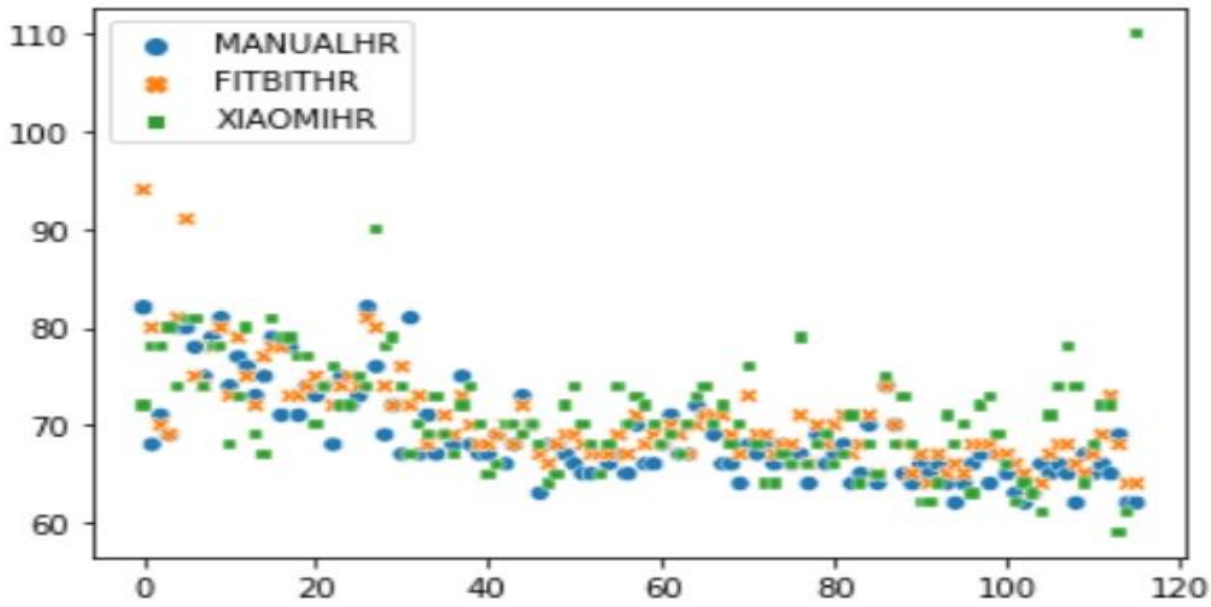


Figure 4.22. Scatter diagram for manual monitor against Fitbit and Xiaomi for heart rate. ***Key:*** ***MANUALHR*** *– Patient Monitor heart rate* ***XIAOMIHR –*** *Xiaomi heart rate.* ***FITBITHR-*** *Fitbit heart rate*

Chapter5 Discussion

The correlation analysis between the wearable devices and the gold standard measurement devices shows a positive correlation for the evaluated Pearson correlation coefficient. Fitbit recorded a strong positive correlation of 0.935949 for heart rate against the patient monitor, which is a very good linear relationship as expected. Xiaomi also recorded a strong positive correlation of 0.749383 for heart rate against the patient monitor. Again, Xiaomi and Fitbit recorded a correlation coefficient of 0.422386 and 0.825244 for heart rate respectively. A comparison between the correlation between the wearable devices and the gold standard devices revealed that the wearable devices have better correlation with the patient monitor than the manual monitoring device.

The Bland-Altman plot for Fitbit heart rate against patient monitor showed a mean difference of 0.63bpm, 0.91bpm and 0.77bpm on day 1, day 2 and overall respectively. This means that the Fitbit device was over estimating the heart rate by 0.63 beats per minute on day 1, 0.91 beats per minutes on day 2 and an overall over estimation of 0.77 beats per minute. The overall heart rate over estimation was not more than even a beat per minute and with a narrow overall limit between -3.78 and 5.33 which implies that the Fitbit device is very accurate and therefore, further validation studies can be made on them and used to supplement the few patient monitors available in the hospitals. The Bland-Altman plot for Xiaomi against the patient monitor showed a mean difference of -1.22bpm on day 1, -0.51bpm on day 2 and an overall mean difference of -0.88bpm. This result means that Xiaomi under estimated the heart rate measurement by 1.22 beats per minutes on day1, 0.51 beats per minutes on day 2 and an overall under estimation of 0.88 beats per minutes. The Bland-Altman plot for Xiaomi also revealed a quite large limit of agreement between -10.53 and 8.82 compared to that of Fitbit. Only 6 and 15 outliers were found in the plot for Fitbit and Xiaomi respectively out of the 237 data points which means that the remaining data points lie in the 95% confidence interval. The Fitbit device used does not measure SpO2 data but the Xiaomi device does. The Bland-Altman plot for Xiaomi SpO2 data against the PHILIPS MX 450 patient monitor SpO2 data revealed a mean difference of 1.11% on day 1, -0.78% on day 2 and an overall of -0.33%. The interpretation to this result is that Xiaomi overestimated SpO2 by 1.11% on day 1, under estimated SpO2 by 0.78% on day 2 and overall, under estimated SpO2 by just 0.33%. The plot also showed a narrow limit of agreement between -3.47 to 2.81, which is so because all of the SpO2 readings recorded were between 95% and 100%. The Bland-Altman plot for Fitbit heart rate data against the manual monitoring device showed that Fitbit was under estimating heart rate measurement

by 1.56 beats per minutes with a level of agreement between -7.23 and 4.11 with 3 outliers. When the results from Fitbit against the patient monitor was compared to Fitbit against the manual monitor, it can be seen that the limits of agreement of Fitbit against the manual monitor is larger than that against the patient monitor hence more accuracy when compared to the patient monitor than when compared to the manual monitor. The Bland-Altman plot for Xiaomi heart rate against the manual monitor displayed a heart rate under estimation by 2.12 beats per minutes and a limit of agreement between -15.07 and 10.83 with only one outlier over a 95% confidence interval. Similar to the case of Fitbit, Xiaomi recorded large limits of agreement when compared to the manual monitor than to the patient monitor that means that Xiaomi has a better accuracy when compared to the patient monitor than to the manual monitoring device.

The accuracy of the devices were measured by a metric called the mean absolute percentage error as explained at the earlier parts of the statistical analysis section. From the MAPE the accuracy of the devices in regards to different vital signs were calculated in percentages. It is very overwhelming to say that the devices have generally very impressive accuracy with regard to the various vital sign measurements. Fitbit recorded a heart rate accuracy measurement of 97.16% on day 1, 96.6% on day 2 and an overall heart rate accuracy of 96.87% when compared to the patient monitor. Xiaomi recorded a heart rate accuracy of 93.85% on day 1, 94.61% on day 2 and an overall accuracy of 94.24% when compared against the patient monitor. Xiaomi again recorded an accuracy for SpO2 measurement against the patient monitor as follows; 98.08% on day 1, 99.01% on day 2 and an overall accuracy of 98.79%. When compared to the manual monitor, Fitbit and Xiaomi recorded an accuracy of 96.64% and 93.21% respectively for heart rate measurement.

The values of accuracy recorded by these devices are very impressive and can therefore be validated with further studies using actual patients for the study.

5.1 Result validation

Similar study conducted in literature obtained error estimation between 1% to 9% and limits of agreement between -27.3 to 13.1 bpm for Fitbit heart rate which is less accurate compared to the values obtained in this study [40]. Another study reported an error estimation of 10.79% for Fitbit heart rate [41]. The study further elaborated that the standard error for heart rate measurement must be within 5% error threshold. Again, the study recorded a correlation coefficient of 0.94 for Fitbit heart rate measurement and was lower during physical activities of the participants. One study in literature reported that an error margin of less than ±10% have

been widely accepted by organizations for wearable devices used for estimating heart rate [42]. The study recorded a mean absolute percentage error of 5.96% and a mean difference of -3.47 bpm for Fitbit heart rate. Although many studies have been conducted on Fitbit, no study has been reported on the use of Xiaomi for vital sign monitoring. Different interpretation of the values of the correlation coefficient have been done in literature with some studies reporting a range >0.5 [43].

All results for error estimations by both Xiaomi and Fitbit in measuring heart rate and SpO2 were less than the 10% standard which means that these wearable devices can be further employed in the measurement of vital signs in hospitals here in Ghana to supplement the patient monitors and hopefully replace the manual monitoring methods.

5.2 Limitations

Even though the project team was able to achieve its primary goal of designing a web application to monitor the vital signs and subsequently perform statistical analysis on the data to evaluate the accuracy of the wearable devices in relation to the gold standards, some limitations were encountered. The limitations are outlined as follows;

1. The study considered only healthy individuals: This was as a result of the inability to obtain ethical clearance to include patients from the hospital. As such, the results of the statistical analysis may not provide sufficient data on both healthy and unhealthy individuals.
2. Selecting the right set of tools for the development of the web application: The design team faced some difficulties in defining the technological stack to employ, which includes determining the Application Programming Interfaces (API's) to use for extraction of data from the Xiaomi wearable device.
3. Lack of sufficient wearable devices to include more volunteers to the study: Due to unavailability of the wearable devices to the team, the number of volunteers for the study was limited to one.

5.3 Conclusion

The two wearable devices used in this study (Fitbit Versa 2 and Xiaomi Mi Band 6) proved to be very accurate in measuring vital signs (SpO2 and heart rate). All two devices achieved values of the mean absolute percentage error (MAPE) to be less than 10% (the generally

accepted value of MAPE). The Fitbit device outperformed the Xiaomi device during heart rate measurement with reference to both the PHILIPS MX 450 patient monitor and manual monitoring device. Only the Xiaomi device has a feature to measure SpO2 and these measurements were very accurate against the patient monitor. The results of the study are in agreement with studies in literature just that this study has shown more accuracy in vital sign measurement using the wearable devices and debunking some studies that reported a decrease in accuracy due to skin tone (high melanin). Results of this study can help in advising the validation and certification of these devices to be used alongside the fully functional web application developed by the team in the hospital setting here in Ghana to monitor vital signs of patients. Further researches should focus on conducting similar study on patients in the hospital setting instead of using healthy individuals.

References

[1] A. Godfrey, V. Hetherington, H. Shum, P. Bonato, N. H. Lovell, and S. Stuart, "From A to Z: Wearable technology explained," *Maturitas*, vol. 113, no. April, pp. 40–47, 2018, doi: 10.1016/j.maturitas.2018.04.012.

[2] M. Di Rienzo, F. Rizzo, G. Parati, G. Brambilla, M. Ferratini, and P. Castiglioni, "MagIC system: A new textile-based wearable device for biological signal monitoring. Applicability in daily life and clinical setting," *Annu. Int. Conf. IEEE Eng. Med. Biol. - Proc.*, vol. 7 VOLS, pp. 7167–7169, 2005, doi: 10.1109/iembs.2005.1616161.

[3] D. Dias and J. P. S. Cunha, "Wearable health devices—vital sign monitoring, systems and technologies," *Sensors (Switzerland)*, vol. 18, no. 8, 2018, doi: 10.3390/s18082414.

[4] M. Buist, S. Bernard, T. V Nguyen, G. Moore, and J. Anderson, "Association between clinically abnormal observations and subsequent in-hospital mortality : a prospective study," vol. 62, pp. 137–141, 2004, doi: 10.1016/j.resuscitation.2004.03.005.

[5] K. M. Hillman, P. J. Bristow, T. Chey, K. Daffurn, T. Jacques, and S. L. Norman, "Antecedents to hospital deaths," no. December 1996, pp. 343–348, 2001.

[6] D. J. Beckett, C. F. Gordon, R. Paterson, S. Chalkley, D. C. Macleod, and D. Bell, "Assessment of clinical risk in the out of hours hospital prior to the introduction of Hospital at Night," vol. 8, no. 1, pp. 33–38, 2009.

[7] Visimobile, "Visi Mobile System." http://www.visimobile.com (accessed May 03, 2022).

[8] R. Karthikamani, P. S. Y. Prasath, M. V. Sree, and J. Sangeetha, "Wireless patient monitoring system," *Int. J. Sci. Technol. Res.*, vol. 8, no. 8, pp. 1081–1084, 2019.

[9] S. H. Limited, "Sensium Health Patch." http://www.sensium.co.uk.

[10] M. Hernandez-Silveira, K. Wieczorkowski-Rettinger, S. Ang, and A. Burdett, "Preliminary assessment of the SensiumVitals®: A low-cost wireless solution for patient surveillance in the general wards," *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. EMBS*, vol. 2015-November, pp. 4931–4937, 2015, doi: 10.1109/EMBC.2015.7319498.

[11] J. Johnson, “A beginners guide to web application development (2021),” Jan. 24, 2020. https://budibase.com/blog/web-application-development/ (accessed Mar. 31, 2022).

[12] “Browser vendors win war with W3C over HTML and DOM standards | ZDNet.” https://web.archive.org/web/20190529021959/https://www.zdnet.com/article/browser-vendors-win-war-with-w3c-over-html-and-dom-standards/ (accessed Oct. 10, 2022).

[13] “HTML5 Differences from HTML4.” https://www.w3.org/TR/html5-diff/ (accessed Oct. 10, 2022).

[14] “Learn to style HTML using CSS - Learn web development | MDN.” https://developer.mozilla.org/en-US/docs/Learn/CSS (accessed Oct. 10, 2022).

[15] “HTML & CSS - W3C.” https://www.w3.org/standards/webdesign/htmlcss#whatcss (accessed Oct. 10, 2022).

[16] “ECMAScript® 2023 Language Specification.” https://tc39.es/ecma262/#sec-overview (accessed Oct. 10, 2022).

[17] “Introduction · Bootstrap v5.0.” https://getbootstrap.com/docs/5.0/getting-started/introduction/ (accessed Oct. 10, 2022).

[18] “Bootstrap 5 Get Started.” https://www.w3schools.com/bootstrap5/bootstrap_get_started.php (accessed Oct. 10, 2022).

[19] “font-awesome - Libraries - cdnjs - The #1 free and open source CDN built to make life easier for developers.” https://cdnjs.com/libraries/font-awesome (accessed Oct. 10, 2022).

[20] “Easy plotting With Chart.js.” https://www.i-programmer.info/news/167-javascript/9719-easy-plotting-with-chartjs.html (accessed Oct. 10, 2022).

[21] “Django documentation | Django documentation | Django.” https://docs.djangoproject.com/en/4.1/ (accessed Oct. 10, 2022).

[22] “Introduction to Django.” https://www.w3schools.com/django/django_intro.php (accessed Oct. 10, 2022).

[23] “SQLite Home Page.” https://www.sqlite.org/index.html (accessed Oct. 10, 2022).

[24] “PostgreSQL - Overview.” https://www.tutorialspoint.com/postgresql/postgresql_overview.htm (accessed Oct. 10, 2022).

[25] “PostgreSQL: Documentation: 14: 1. What Is PostgreSQL?” https://www.postgresql.org/docs/current/intro-whatis.html (accessed Oct. 10, 2022).

[26] “Visual Studio Code Frequently Asked Questions.” https://code.visualstudio.com/docs/supporting/faq (accessed Oct. 10, 2022).

[27] “Fitbit.” https://www.fitbit.com/dev (accessed Oct. 10, 2022).

[28] “Web API.” https://dev.fitbit.com/build/reference/web-api/ (accessed Oct. 10, 2022).

[29] ““Huami provides Web API for accessing user activity data tracked with Huami wearable-devices. | by Diogo Kamioka | Medium.” https://dkamioka.medium.com/huami-provides-web-api-for-accessing-user-activity-data-tracked-with-huami-wearable-devices-cfd7351610f4 (accessed Oct. 10, 2022).

[30] “What is Heroku | Heroku.” https://www.heroku.com/what (accessed Oct. 10, 2022).

[31] “Installation — Psycopg 2.9.4 documentation.” https://www.psycopg.org/docs/install.html (accessed Oct. 10, 2022).

[32] “Installation — Gunicorn 20.1.0 documentation.” https://docs.gunicorn.org/en/latest/install.html (accessed Oct. 10, 2022).

[33] “Configuring Django Apps for Heroku | Heroku Dev Center.” https://devcenter.heroku.com/articles/django-app-configuration (accessed Oct. 10, 2022).

[34] “WhiteNoise 6.2.0 documentation.” http://whitenoise.evans.io/en/stable/ (accessed Oct. 10, 2022).

[35] M. Nissen *et al.*, “Heart Rate Measurement Accuracy of Fitbit Charge 4 and Samsung Galaxy Watch Active2: Device Evaluation Study,” *JMIR Form. Res.*, vol. 6, no. 3, pp. 1–14, 2022, doi: 10.2196/33635.

[36] K. Rani Das, “A Brief Review of Tests for Normality,” *Am. J. Theor. Appl. Stat.*, vol. 5, no. 1, p. 5, 2016, doi: 10.11648/j.ajtas.20160501.12.

[37] P. Ahlgren, B. Jarneving, and R. Rousseau, “Requirements for a cocitation similarity measure, with special reference to Pearson’s correlation coefficient,” *J. Am. Soc. Inf. Sci. Technol.*, vol. 54, no. 6, pp. 550–560, 2003, doi: 10.1002/asi.10242.

[38] H. Lei and I. Vorechovsky, “Haixin Lei 1 Jan Zaloudik 2 Igor Vorechovsky 1*,” *Clin. Chem.*, no. 5, pp. 799–801, 2002.

[39] A. de Myttenaere, B. Golden, B. Le Grand, and F. Rossi, “Mean Absolute Percentage Error for regression models,” *Neurocomputing*, vol. 192, pp. 38–48, 2016, doi: 10.1016/j.neucom.2015.12.114.

[40] M. P. Wallen, S. R. Gomersall, S. E. Keating, U. Wisløff, and J. S. Coombes, “Accuracy of heart rate watches: Implications for weight management,” *PLoS One*, vol. 11, no. 5, pp. 1–9, 2016, doi: 10.1371/journal.pone.0154420.

[41] R. Kondama Reddy *et al.*, “Accuracy of wrist-worn activity monitors during common daily physical activities and types of structured exercise: Evaluation study,” *JMIR mHealth uHealth*, vol. 6, no. 12, 2018, doi: 10.2196/10338.

[42] B. W. Nelson and N. B. Allen, “Accuracy of consumer wearable heart rate measurement during an ecologically valid 24-hour period: Intraindividual validation study,” *JMIR mHealth uHealth*, vol. 7, no. 3, pp. 1–16, 2019, doi: 10.2196/10828.

[43] F. El-Amrawy and M. I. Nounou, “Are currently available wearable devices for activity tracking and heart rate monitoring accurate, precise, and medically beneficial?,” *Healthc. Inform. Res.*, vol. 21, no. 4, pp. 315–320, 2015, doi: 10.4258/hir.2015.21.4.315.

Appendix

Codes for Data Analysis

```
stats.probplot(data["PTMTHR"], dist = "norm", plot = pylab)
pylab.show()
sns.distplot (data["PTMTHR"])
shapiro(data["PTMTHR"])
sns.distplot(data["XIAOMIHR"])

shapiro(data["XIAOMIHR"])
stats.probplot(data["FITBITHR"], dist = "norm", plot = pylab)
pylab.show()
sns.distplot(data["FITBITHR"])
shapiro(data["FITBITHR"])
stats.spearmanr(data["PTMTHR"], data["FITBITHR"])
import statsmodels.api as sm
f, ax = plt.subplots(1, figsize =(8,5))
sm.graphics.mean_diff_plot(data["PTMTHR"], data["XIAOMIHR"],ax =ax)
plt.xlabel("Mean")
plt.ylabel("Difference")
plt.show()

import statsmodels.api as sm
f, ax = plt.subplots(1, figsize =(8,5))
sm.graphics.mean_diff_plot(data["PTMTHR"], data["FITBITHR"],ax =ax)
plt.xlabel("Mean")
plt.ylabel("Difference")
plt.show()

import pandas as pd
SPO2 = pd.read_csv('Overall_SPO2_data.csv')
import pylab
import scipy.stats as stats
stats.probplot(SPO2["PTMT"], dist = "norm", plot = pylab)
pylab.show()

from scipy.stats import shapiro
shapiro(SPO2["PTMT"])
import seaborn as sns
sns.distplot(SPO2["PTMT"])

import pylab
import scipy.stats as stats
stats.probplot(SPO2["XIAOMI"], dist = "norm", plot = pylab)
pylab.show()

shapiro(SPO2['XIAOMI'])
sns.distplot(SPO2["XIAOMI"])
```

```
# Define the dataset as python lists
actual   = SPO2["PTMT"]
forecast = SPO2["XIAOMI"]

# Consider a list APE to store the
# APE value for each of the records in dataset
APE = []

# Iterate over the list values
for day in range(151):

    # Calculate percentage error
    per_err = (actual[day] - forecast[day]) / actual[day]

    # Take absolute value of
    # the percentage error (APE)
    per_err = abs(per_err)

    # Append it to the APE list
    APE.append(per_err)

# Calculate the MAPE
MAPE = sum(APE)/len(APE)

# Print the MAPE value and percentage
print(f'''
MAPE   : { round(MAPE, 2) }
MAPE % : { round(MAPE*100, 2) } %
''')

# Define the dataset as python lists
actual   = data["PTMTHR"]
forecast = data["FITBITHR"]

# Consider a list APE to store the
# APE value for each of the records in dataset
APE = []

# Iterate over the list values
for day in range(237):

    # Calculate percentage error
    per_err = (actual[day] - forecast[day]) / actual[day]

    # Take absolute value of
```

```
    # the percentage error (APE)
    per_err = abs(per_err)

    # Append it to the APE list
    APE.append(per_err)

# Calculate the MAPE
MAPE = sum(APE)/len(APE)

# Print the MAPE value and percentage
print(f'''
MAPE   : { round(MAPE, 2) }
MAPE % : { round(MAPE*100, 2) } %
''')



# Define the dataset as python lists
actual   = data["PTMTHR"]
forecast = data["XIAOMIHR"]

# Consider a list APE to store the
# APE value for each of the records in dataset
APE = []

# Iterate over the list values
for day in range(237):

    # Calculate percentage error
    per_err = (actual[day] - forecast[day]) / actual[day]

    # Take absolute value of
    # the percentage error (APE)
    per_err = abs(per_err)

    # Append it to the APE list
    APE.append(per_err)

# Calculate the MAPE
MAPE = sum(APE)/len(APE)

# Print the MAPE value and percentage
print(f'''
MAPE   : { round(MAPE, 3) }
MAPE % : { round(MAPE*100, 2) } %
''')

import statsmodels.api as sm
```

```
import matplotlib.pyplot as plt
f, ax = plt.subplots(1, figsize =(8,5))
sm.graphics.mean_diff_plot(SPO2["PTMT"], SPO2["XIAOMI"],ax =ax)
plt.xlabel("Mean")
plt.ylabel("Difference")
plt.show()

day1= pd.read_csv("Heart_rate_data_day1.csv")
import statsmodels.api as sm
f, ax = plt.subplots(1, figsize =(8,5))
sm.graphics.mean_diff_plot(day1["PTMTHR"], day1["XIAOMIHR"],ax =ax)
plt.xlabel("Mean")
plt.ylabel("Difference")
plt.show()

import statsmodels.api as sm
f, ax = plt.subplots(1, figsize =(8,5))
sm.graphics.mean_diff_plot(day2["PTMTHR"], day2["FITBITHR"],ax =ax)
plt.xlabel("Mean")
plt.ylabel("Difference")
plt.show()
day2 = pd.read_csv("Heart_rate_data_day1.csv")

day2
import statsmodels.api as sm
f, ax = plt.subplots(1, figsize =(8,5))
sm.graphics.mean_diff_plot(day2["PTMTHR"], day2["XIAOMIHR"],ax =ax)
plt.xlabel("Mean")
plt.ylabel("Difference")
plt.show()

import statsmodels.api as sm
f, ax = plt.subplots(1, figsize =(8,5))
sm.graphics.mean_diff_plot(day1["PTMTHR"], day1["FITBITHR"],ax =ax)
plt.xlabel("Mean")
plt.ylabel("Difference")
plt.show()
spo2day1 = pd.read_csv("SPO2_data_day1.csv")

import statsmodels.api as sm
f, ax = plt.subplots(1, figsize =(8,5))
sm.graphics.mean_diff_plot(spo2day1["PTMT"], spo2day1["XIAOMI"],ax =ax)
plt.xlabel("Mean")
plt.ylabel("Difference")
plt.show()

spo2day2 = pd.read_csv("SPO2_data_day2.csv")
import statsmodels.api as sm
```

```
f, ax = plt.subplots(1, figsize =(8,5))
sm.graphics.mean_diff_plot(spo2day2["PTMT"], spo2day2["XIAOMI"],ax =ax)
plt.xlabel("Mean")
plt.ylabel("Difference")
plt.show()


myspo2 = pd.read_csv("Overall_spo2_data.csv")
myspo2
import statsmodels.api as sm
f, ax = plt.subplots(1, figsize =(8,5))
sm.graphics.mean_diff_plot(myspo2["PTMT"], myspo2["XIAOMI"],ax =ax)
plt.xlabel("Mean")
plt.ylabel("Difference")
plt.show()


# Define the dataset as python lists
actual   = SPO2["PTMT"]
forecast = SPO2["XIAOMI"]

# Consider a list APE to store the
# APE value for each of the records in dataset
APE = []

# Iterate over the list values
for day in range(151):

    # Calculate percentage error
    per_err = (actual[day] - forecast[day]) / actual[day]

    # Take absolute value of
    # the percentage error (APE)
    per_err = abs(per_err)

    # Append it to the APE list
    APE.append(per_err)

# Calculate the MAPE
MAPE = sum(APE)/len(APE)

# Print the MAPE value and percentage
print(f'''
MAPE   : { round(MAPE, 2) }
MAPE % : { round(MAPE*100, 2) } %
''')
# Define the dataset as python lists
actual   = spo2day1["PTMT"]
```

```
forecast = spo2day1["XIAOMI"]

# Consider a list APE to store the
# APE value for each of the records in dataset
APE = []

# Iterate over the list values
for day in range(36):

    # Calculate percentage error
    per_err = (actual[day] - forecast[day]) / actual[day]

    # Take absolute value of
    # the percentage error (APE)
    per_err = abs(per_err)

    # Append it to the APE list
    APE.append(per_err)

# Calculate the MAPE
MAPE = sum(APE)/len(APE)

# Print the MAPE value and percentage
print(f'''
MAPE   : { round(MAPE, 2) }
MAPE % : { round(MAPE*100, 2) } %
''')

# Define the dataset as python lists
actual   = spo2day2["PTMT"]
forecast = spo2day2["XIAOMI"]

# Consider a list APE to store the
# APE value for each of the records in dataset
APE = []

# Iterate over the list values
for day in range(115):

    # Calculate percentage error
    per_err = (actual[day] - forecast[day]) / actual[day]

    # Take absolute value of
    # the percentage error (APE)
    per_err = abs(per_err)

    # Append it to the APE list
    APE.append(per_err)
```

```
# Calculate the MAPE
MAPE = sum(APE)/len(APE)

# Print the MAPE value and percentage
print(f'''
MAPE   : { round(MAPE, 2) }
MAPE % : { round(MAPE*100, 2) } %
''')
# Define the dataset as python lists
actual   = data["PTMTHR"]
forecast = data["XIAOMIHR"]

# Consider a list APE to store the
# APE value for each of the records in dataset
APE = []

# Iterate over the list values
for day in range(237):

    # Calculate percentage error
    per_err = (actual[day] - forecast[day]) / actual[day]

    # Take absolute value of
    # the percentage error (APE)
    per_err = abs(per_err)

    # Append it to the APE list
    APE.append(per_err)

# Calculate the MAPE
MAPE = sum(APE)/len(APE)

# Print the MAPE value and percentage
print(f'''
MAPE   : { round(MAPE, 2) }
MAPE % : { round(MAPE*100, 2) } %
''')

# Define the dataset as python lists
actual   = data["PTMTHR"]
forecast = data["FITBITHR"]

# Consider a list APE to store the
# APE value for each of the records in dataset
APE = []

# Iterate over the list values
```

```
for day in range(237):

    # Calculate percentage error
    per_err = (actual[day] - forecast[day]) / actual[day]

    # Take absolute value of
    # the percentage error (APE)
    per_err = abs(per_err)

    # Append it to the APE list
    APE.append(per_err)

# Calculate the MAPE
MAPE = sum(APE)/len(APE)

# Print the MAPE value and percentage
print(f'''
MAPE   : { round(MAPE, 2) }
MAPE % : { round(MAPE*100, 2) } %
''')

# Define the dataset as python lists
actual   = day1["PTMTHR"]
forecast = day1["FITBITHR"]

# Consider a list APE to store the
# APE value for each of the records in dataset
APE = []

# Iterate over the list values
for day in range(116):

    # Calculate percentage error
    per_err = (actual[day] - forecast[day]) / actual[day]

    # Take absolute value of
    # the percentage error (APE)
    per_err = abs(per_err)

    # Append it to the APE list
    APE.append(per_err)

# Calculate the MAPE
MAPE = sum(APE)/len(APE)

# Print the MAPE value and percentage
print(f'''
MAPE   : { round(MAPE, 2) }
```

```
MAPE % : { round(MAPE*100, 2) } %
''')

# Define the dataset as python lists
actual   = day1["PTMTHR"]
forecast = day1["XIAOMIHR"]

# Consider a list APE to store the
# APE value for each of the records in dataset
APE = []

# Iterate over the list values
for day in range(116):

    # Calculate percentage error
    per_err = (actual[day] - forecast[day]) / actual[day]

    # Take absolute value of
    # the percentage error (APE)
    per_err = abs(per_err)

    # Append it to the APE list
    APE.append(per_err)

# Calculate the MAPE
MAPE = sum(APE)/len(APE)

# Print the MAPE value and percentage
print(f'''
MAPE   : { round(MAPE, 2) }
MAPE % : { round(MAPE*100, 2) } %
''')
# Define the dataset as python lists
actual   = day2["PTMTHR"]
forecast = day2["XIAOMIHR"]

# Consider a list APE to store the
# APE value for each of the records in dataset
APE = []

# Iterate over the list values
for day in range(121):

    # Calculate percentage error
    per_err = (actual[day] - forecast[day]) / actual[day]

    # Take absolute value of
    # the percentage error (APE)
```

```
    per_err = abs(per_err)

    # Append it to the APE list
    APE.append(per_err)

# Calculate the MAPE
MAPE = sum(APE)/len(APE)

# Print the MAPE value and percentage
print(f'''
MAPE   : { round(MAPE, 2) }
MAPE % : { round(MAPE*100, 2) } %
''')

# Define the dataset as python lists
actual   = day2["PTMTHR"]
forecast = day2["FITBITHR"]

# Consider a list APE to store the
# APE value for each of the records in dataset
APE = []

# Iterate over the list values
for day in range(121):

    # Calculate percentage error
    per_err = (actual[day] - forecast[day]) / actual[day]

    # Take absolute value of
    # the percentage error (APE)
    per_err = abs(per_err)

    # Append it to the APE list
    APE.append(per_err)

# Calculate the MAPE
MAPE = sum(APE)/len(APE)

# Print the MAPE value and percentage
print(f'''
MAPE   : { round(MAPE, 2) }
MAPE % : { round(MAPE*100, 2) } %
''')

import seaborn as sns
sns.scatterplot(data= data)
sns.lineplot(data= data)
```