



## **ASHESI UNIVERSITY**

**A UNIVERSAL PLATFORM FOR ELECTRONIC HEALTH RECORDS FOR LOW  
RESOURCE HOSPITALS**

**CAPSTONE PROJECT**

B.Sc. Computer Engineering

**TRACI EWURABENA NYHIRA GYEBI**

**2022**

**ASHESI UNIVERSITY**

**A UNIVERSAL PLATFORM FOR ELECTRONIC HEALTH RECORDS FOR LOW  
RESOURCE HOSPITALS**

**CAPSTONE PROJECT**

Capstone Project submitted to the Department of Engineering, Ashesi University in  
partial fulfilment of the requirements for the award of Bachelor of Science degree in  
Computer Engineering.

**TRACI EWURABENA NHYIRA GYEBI**

**2022**

**DECLARATION**

I hereby declare that this capstone is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

.....

Candidate's Name:

.....

Date:

.....

I hereby declare that preparation and presentation of this capstone were supervised in accordance with the guidelines on supervision of capstone laid down by Ashesi University.

Supervisor's Signature:

.....

Supervisor's Name:

.....

Date:

.....

## **ACKNOWLEDGEMENTS**

I would like to express my gratitude to the Almighty God for seeing me through my four years at Ashesi University. Without a doubt, I am grateful to Dr. Elena Rosca, my supervisor, for her attention, passion, commitment, and continuous support. I would also like to express my sincere gratitude to Elvis Segbawu, and Jeremiah Paul Takyi for their unwavering support throughout the project. Finally, I would like to express my gratitude to my family and friends for encouraging me to reach this significant milestone. God bless you all.

## **ABSTRACT**

Due to the lack of health management systems and technical know-how, most hospitals in Ghana continue to use a file-based system approach. While this method works, it is ineffective for storing and managing medical records. It is difficult to organize and maintain track of information that's dispersed all over the place. Likewise, there is no smart system in place to help transfer medical data from an instrument into the database system of a hospital management to aid in efficiency. Hence, this paper examines the shortcomings in the management methods utilized by Ghana's healthcare institutions, as well as the inefficiencies that result. It goes into details on the nature of electronic health records and the benefits they provide, such as making distance communication and data interchange more accessible, improving the quality of medical treatment, and so on. The project is broken down into three sections. The project's initial goal is to create a hospital management system that eliminates the need for paper medical records. Followed by the transfer of patient data from one hospital to another however, the hospitals must be on the same management system. Finally, to aid in making the system smart is the creation a pulse oximeter that sends the SpO<sub>2</sub> value to the hospital management system's database. The report finishes by elaborating on the project's challenges, limitations, and a recommendation for what to do next.

## Table of Contents

<b>DECLARATION</b> .....	<b>i</b>
<b>ACKNOWLEDGEMENTS</b> .....	<b>ii</b>
<b>ABSTRACT</b> .....	<b>iii</b>
<b>Chapter 1: Introduction</b> .....	<b>1</b>
1.1 Introduction .....	1
1.2 Problem Statement.....	3
1.3 Motivation .....	3
1.4 Proposed Solution.....	4
1.5 Expected Outcome.....	5
<b>Chapter 2: Literature Review</b> .....	<b>6</b>
2.1. Introduction .....	6
2.2 Related Works .....	9
2.3 Gaps and Improvements .....	11
<b>Chapter 3: Design Requirements</b> .....	<b>13</b>
3.1 Overview .....	13
3.2 Design Objective .....	13
3.3 Requirement Gathering & Analysis .....	14
3.4 User Identification .....	14
3.5 System Overview.....	14

3.5.1 Functional Requirement of the Hospital Management System .....	14
3.5.2 Non-Functional Requirement of the Hospital Management System.....	17
3.6 Component Selection.....	17
3.6.1 Pugh Matrix .....	17
3.6.2 Final Components.....	18
3.7 Software System Design.....	19
3.7.1 Hospital Database .....	20
3.7.2 Pulse Oximeter Data.....	20
3.7.3 Transfer Patient Data .....	21
3.8 Hardware System Design .....	22
3.8.1 The Pulse Oximeter .....	22
<b>CHAPTER 4: Implementation .....</b>	<b>24</b>
4.1 Hospital Management System Implementation.....	24
4.1.1 Hospital Database .....	24
4.1.2 Transfer Patient Data .....	41
4.1.3 Pulse Oximeter .....	43
4.1.3.1 How the pulse oximeter operates.....	43
4.1.3.2 How the circuitry was built .....	43
4.1.3.3 Programming and calculation of the pulse oximeter .....	44
4.1.3.4 Transferring the Data unto the HMS' Database .....	46

<b>CHAPTER 5: Testing</b> .....	<b>48</b>
5.1 Overview .....	48
5.2 Transfer of Patient Data Test.....	48
5.3 Hospital Database Test .....	51
5.3.1 Administration Test .....	52
5.3.2 Receptionist Test .....	54
5.3.3 Nurse Test.....	55
5.3.4 Doctor Test .....	56
5.3.5 Lab Technician Test .....	57
5.3.6 Pharmacist Test.....	58
5.4 Pulse Oximeter Test.....	59
<b>CHAPTER 6: Conclusion</b> .....	<b>62</b>
6.1 Overview .....	62
6.2 Challenges .....	62
6.3 Future Works .....	62
6.4 Conclusion.....	63
<b>References</b> .....	<b>65</b>



# Chapter 1: Introduction

## 1.1 Introduction

Five years ago, the healthcare and medical industries were very different. This has been altered because of the technological evolutions that have enabled medical advancements [1]. It is argued that the future of healthcare rests in collaborating with technology, and healthcare workers must embrace evolving healthcare technologies to remain relevant in the following years [2]. Technology has had a significant impact on the medical industry. The medical industry can achieve significant paradigm shifts in treatment, data collection, remedy research, and many other areas by utilizing technology. Due to technological improvements, the medical industry is now very accessible to the general population. With technology, medical and health professionals may provide better care to patients and treat illnesses [1].

Despite technological advancements, the healthcare system in Ghana is underutilizing the technology by automating customer records fully. Ghana's healthcare system is unable to keep track of the patients' medical information effectively. Certain hospitals, particularly those in rural areas, still maintain records manually (paperwork) due to low resources. To compound the difficulty, when a patient is transferred from one hospital to another, there is no suitable protocol to speed up the transfer procedure. In a recent conversation with a doctor, it was brought to my attention that patients transferred to other hospitals have their medical records written on paper, even in this pandemic season. Not to say paperwork is unacceptable, but in our current dispensation, it is unsafe to be using such a medium that can lead to the spread of the COVID-19 virus.

Along with the inherent difficulties of maintaining patient data in written form, there is no reliable and quick way to see patient data if the patient attends a regularly frequented hospital. However, with the Electronic Health Records, this extra step will not be required of the patient as the patient's medical record can be transferred via a faster and more efficient medium. However, with the Electronic Health Records, this extra step will not be required of the patient as the medical record of the patient can be transferred via a faster and more efficient medium. Not to end there, even with the transmission of drugs, patients must take written medical reports to the nearest pharmacy before being administered drugs. Imagine the dilemma if the pharmacist cannot see the doctor's report.

The primary goal of this project is to research, investigate, and design a prototype of a universal platform that low-resource hospitals can utilize in Ghana to help with the rapid sharing of medical records from one to another. This platform seeks to make distance communication and data exchange more accessible, improve medical treatment quality, and eliminate the outdated paper-based approach. In the era of the COVID-19 pandemic, it is expected that all work processes that make use of paperwork will be automated to reduce the spread of the virus. Furthermore, while the solution concentrates on the internal operations of a hospital, it also consolidates all institutions and allows them to share medical records. Hospitals can better communicate with their patients, enhance the speed with which medical records are transferred, and raise patient satisfaction with such a dedicated system in place. In a public lecture at Ashesi University, the Vice President of the Republic of Ghana declared that the government is changing the state of healthcare delivery in Ghana, that is sharing the same sentiments as the project. He stated that the record-keeping structure at health facilities across the country has been digitized for more efficient management.

All teaching hospitals and regional hospitals, on the other hand, will be connected and made to communicate with one another as part of the strategy [15].

## **1.2 Problem Statement**

Paper Medical Records are a type of medical record frequently utilized by low-resource hospitals for various reasons, including lack of resources, technical know-how, and many others. Though the paperwork approach may seem outdated, it does offer some advantages such as reduced cost, the ability to customize, and the ease of use, among others; however, given the current state of the world, it is not advisable. Aside from the health challenges it portrays, it can be readily destroyed by floods and other natural catastrophes [3]. Hence, because of their physical limitations, the documents are unable to be exchanged in times of crisis. Healthcare facilities would have no way of exchanging data in an emergency to check important information on a patient before administering medicine. Such hospitals are also unable to handle reviewing raw datasets to detect trends, draw conclusions, and identify prospects for changes, despite technological advancements [4]. The healthcare industry is said to generate massive amounts of data rapidly. Every day, health care facilities around the country generate billions of pieces of information. Furthermore, many health documents are stored in these facilities' archives and records sections. With all of this going on in the country, the question becomes: who is responsible for ensuring the accessibility of the country's billions of health records, and what is the best architecture to manage this?

## **1.3 Motivation**

Technology appears to be rapidly growing in our modern era, and it has become an integral part of our daily lives. Things like storing massive amounts of data, which would take humans hours to accomplish, can be quickly accomplished by computers in the least amount of time. Technology

improves efficiency by handling all the processing and providing users with the information they need to make the best decision possible. Health practitioners will not waste time storing patient data, attending to a patient, and many other tasks if such a medium is available and utilized. Instead, they will be able to focus on the most critical aspects of the patient's care while computers assist in properly medical prescriptions, reducing the chances of errors.

#### **1.4 Proposed Solution**

The proposed solution creates a prototype for a universal platform that can be utilized by both public and private hospitals in Ghana to facilitate the transfer of medical records from one hospital to the other. This system will allow health care practitioners to exchange data (patient medical records) to aid in the diagnosis and administration of appropriate and timely medical therapy. This system will also examine observable raw datasets to spot trends and develop conclusions. This will be accomplished by implementing a pulse oximeter to demonstrate proof of principle.

Finally, some of the system's non-functional requirements is to aid with performance, security, and efficiency. As a result, Ghana cards can be scanned into the system, allowing valuable data from the card to be transferred to the individual's database. Hence, without the patient's Ghana card, certain details will not be disclosed to the patient until the proof of the card.

Some of the advantages of developing and implementing this unified health information exchange system for its different stakeholders include:

1. It ensures that data is transferred in real-time with the patient's consent.
2. Providing accurate, up-to-date, and comprehensive information to patients.
3. Improving the safety and dependability of prescription.
4. Securely sharing electronic information with patients and other physicians.

5. Improving patient and provider engagement and communication, as well as healthcare convenience.

### **1.5 Expected Outcome**

The project should make it easier for Ghana's healthcare organizations to share patient data. The interested parties would access the client's information in the shortest feasible period by making a simple request through the web application. Insights can also be acquired from current and previous patient data to aid a doctor's decision-making through data prediction. Lastly, to quicken the data inputting process, the Ghana Card which is the most used and reliable card in Ghana currently will be scanned using the QR scanner and uploaded to the database directly [5].

## **Chapter 2: Literature Review**

### **2.1. Introduction**

The idea of a Hospital Information System has all the earmarks of being the reconciliation and expansiveness of patient or Health Information Systems (HIS). To be called a HIS, a framework should have the option to address the issues of those it will serve from one side of the planet to the other. If the medical clinic is the HIS's client in the present circumstance, the HIS should give worldwide and departmental data on the emergency clinic's state. Also, the HIS data set should contain all clinical data about the patient, empowering for patient information revealing and the board across every single clinical office and information sources [6]. A HIS can be planned in various architectural styles. It is achieved by connecting various departmental or clinical data frameworks to a focal framework. Notwithstanding several worldwide applications, a second technique that has been developed is to have departmental or clinical framework applications. Due to the requirements of every current framework, any current far reaching HIS will be a blend of departmental/clinical framework interfaces and the applications/information base of the HIS bought by the emergency clinic.

The following are key features to include in an HIS:

1. Patient information databases
2. Obtaining patient information
3. Admission of patients and bed control
4. Applications for management and evaluation of patients
5. Computer-aided decision-making

HIS also refers to the gathering, analyzing, storing, and safeguarding of high-quality patient health data [7]. Information can be collected on paper, in a combination of paper and digital (hybrid), or

in an Electronic Health Record [7]. An electronic health record (EHR) is a digitally recorded systematized collection of patient and population health information. The history of the HIS sector may be traced back to the 1920s, when medical professionals discovered the value of having written forms of patient data [8]. Paper-based patient data documentation was the predominant method of data storage in the pre-digital era.

The Mayo Clinic in Rochester, Minnesota, was one of the main significant wellbeing frameworks to execute an Electronic Health Record (EHR) during the 1960s. EHRs were so costly during the 1960s that the public authority solely utilized them as a team with wellbeing associations. Simply the biggest emergency clinics could involve them during the 1970s, and they were utilized for charging and booking. During the 1960s, another way to deal with clinical records (still on paper) started to arise all the while. This method, known as the "problem-oriented" clinical record, added more fundamental data about patients and developed into the clinical records we see today [16]. Clinicians started gathering and keeping information viewing a patient's set of experiences as a component of the "problem-oriented" clinical record. Whenever appropriately carried out, this idea considered more powerful correspondence among individuals from medical care groups and more straightforward coordination of protection care and upkeep. EHRs were planned with fields that could be filled in with clinical data and saved as an electronic record as EHRs developed more reasonable and accessible during the 1980s. PCs had advanced into most clinical workplaces by the 1990s, and they were being utilized for record-keeping purposes. EHRs were for the most part found in scholastic ongoing and short-term clinical offices, and they included information trade for claims handling and report gathering by means of picture checking. Huge scope change, then again, didn't end up being unmistakable until the web period. Indeed, even in its beginning phases, the web ended up being a productive apparatus for catching and sharing solution narratives and

other clinical records. At long last, all through the previous ten years, most significant clinical frameworks in the industrialized world had the option to convey depending on the situation [9].

The EHR is important for a broader framework that incorporates information section and show, stockpiling, and admittance to the clinical chief, which is generally a specialist or a medical caretaker. To enter data, information is placed by means of a console, correspondence and record, discourse acknowledgment and translation, light pen, contact screen, hand-held electronic notebook, or a hand-held individual advanced colleague (maybe remote) with motion, character acknowledgment, and gathering abilities. Direct instrumentation from electronic patient monitors and bedside terminals, nursing stations, bar code readers, Radio-Frequency Identification (RFID), investigations by other connected PC frameworks, for example, lab autoanalyzer's and emergency unit, or even a solid organization association with another supplier's EHRs are altogether instances of conceivable section strategies [18].

The EHR turns out to be considerably more remarkable when matched with a framework that recovers important clinical information to upgrade clinical direction, increment patient wellbeing, and support quality improvement. EHRs have been displayed to impact doctor conduct and patient results. The basic role of the EHR is to aid the conveyance of clinical treatment to a particular patient. In an ideal world, the EHR satisfies this capacity by conveying authentic and current data about a patient to the doctor, working with correspondence among medical services suppliers about that patient's therapy, and recording the consideration interaction and the rationale behind the choices taken. To improve information accuracy and practicality of decision support, the information in an EHR should be assembled as part of the normal medical care delivery process by the providers of care and their organizations. Furthermore, this information should be disclosed



to aid the patient's decision-making, possibly with the patient's consent or advice to ensure protection [10].

With the rise of the Internet of Things, the medical care industry will have substantially more notable potential. The use of smart devices and sensors to collect data from patients drives specialized advance leaps greatly, because recorded data will no longer be directly placed into the framework but will instead be naturally stacked into the right data set. Sensors open a new front in which specialists and medical attendants can continuously monitor a patient's status without having to wait for them to be in the same room, and they can also be alerted when vitals have spiked or flooded by the sensors' actuators. This improvement also allows for better monitoring and control of the various areas of the emergency clinic, removing potentially dangerous situations such as ambiguity about the number of available beds and maintaining the facility's overall temperature [17].

## **2.2 Related Works**

A hospital management system's capacity to handle information concerning resource availability is critical. Following occupancy rates, calculating the number of required representatives, and assessing the number of available representatives and other assets such as divisions, bed accessibility, administrations, clinical hardware, drugs, and numerous others are all part of a system that deals with the emergency clinic's status and asset accessibility. Such information is crucial in an emergency, and the software that interacts with it should communicate in a common language. For this goal, several standards have been developed.

The SAHANA Disaster Management System (DMS) [11] was one of the first software. The Sahana DMS framework was sent in Haiti in 2010 during the seismic calamity, especially in Port-

au-Prince. By sending data in regards to medical clinic accessibility with crisis responders, this innovation helped in dealing with the progression of casualties in Haiti.

Liapis et al. [13] described how, as part of the IMPRESS project, they developed a Hospital Availability management system that allows hospitals and other healthcare institutions to communicate information about facilities and resources. The data on hospital availability is entered by the hospital operators, who then convey it to the crisis center and first responders as bed, staff, and service availability. In this case, operators are commonly contacted by another hospital or an emergency call center, and they answer by announcing the facility's availability.

The WHO and Global Health Cluster's Health Resources Availability Mapping System (HeRAMS) [13] is another model. Its motivation is to assess the administrations and assets accessible in clinics where there is a wellbeing emergency or crisis. The system depends on clinic reviews that gather information on the accessibility of wellbeing assets and administrations like staff, beds, clinical hardware, and drugs. The overview results are imagined on an intuitive dashboard that shows the present status of emergency clinic assets. The WHO creates logical reports considering the discoveries as a team with nearby wellbeing services to design future endeavors to advance the circumstance. Thus, states can utilize this infrastructure to assist with giving medical care during emergencies [19].

The application of data science in enhancing the Patient Management System is one of the relevant works in data analytics employing a logistic regression model to forecast the risk of patients skipping appointments while optimizing the restrictions that cause them to be missed. Because the length of the wait was one of the main issues for most patients, their approach likewise focused on short wait times [14].

### **2.3 Gaps and Improvements**

In recent years, students and other professionals have constructed database management solutions to help solve this challenge. However, either the project is started and abandoned, or it fails to complete its purpose. As a result of extensive research, this project is focused on addressing hospital data management systems and assisting in the conclusion and creation of trends as to how certain patients have oxygen and blood pressure related diseases. During the project, more research will be conducted, and the data collected will be thoroughly reviewed to help find trends and solve the problem.

To create authentic trends and draw conclusions, a pulse oximeter will be built to measure the oxygen level of the patients especially under the current conditions, it is very important to monitor the breathing rate and oxygen level. After a couple of data has been stored unto the database, the system will be able to conclude on the type of condition an individual has and advise the patient on what to do.

Aside from the benefits of using an electronic health record, it has been noted as having various drawbacks that may stymie development over time. Patients, doctors, and nurses have been shown to spend more time in front of screens and other gadgets due to the EHR, which damages their eyesight in the long run. As a result, the Ghana Card will be used to aid in streamlining the process of data collection. Once the card is scanned, the data of the respective patient will be entered into the database and the subsequent direction will be taken afterwards. This is to help ease the queues in low resource hospitals and help make the hospital experience more convenient. In addition, the government of Ghana has chosen to link the Ghana card to almost all our identities in order to make the country more digital. This includes our SIM cards, passports, and cards for the National

Health Insurance Scheme, among other things. This will help with security, activity tracking and monitoring, and improving customer and employee interactions, and many others.

The database management system will also be built so that hospitals using the same system will be able to transfer medical records to one another on demand, and passwords will be required at required viewpoint levels before the information reaches the respective controllers to prevent a man in the middle attack.

## **Chapter 3: Design Requirements**

### **3.1 Overview**

The qualities that the system would bring to Ghanaian health institutions are examined in depth in this chapter. This project entails the creation of a health management information system that hospitals can use to make better decisions, as previously indicated. As a result, the requirements that this project aims to meet would be provided directly by the initiative's target audience. The purpose of this online hospital management system is to help manage patient's information, aid in the swift transfer of medical records for hospitals on the same medium and to be able to make a smart instrument that will be able to communicate directly with the hospital's management system. The database management system is to be used mainly by low resource hospitals and this is to help curb the use of hardcopies or the paper medical records. The system is based on a relational database with its hospital management and appointment functions. This project will also include the use of sensors to monitor patients' vitals, which will be directly integrated with the hospital management system, allowing seamless input of data and ease of sharing with other hospitals. The patient's Ghana card will be linked to the system's database to help speed up the data entry procedure and secure data security.

### **3.2 Design Objective**

For the design part of this project, the following are the objectives:

1. Design an interoperable hospital system that allows for the rapid transfer of patient data.
2. Design a smart device that measures a patient's oxygen level and uploads the data to the hospital's database.

### **3.3 Requirement Gathering & Analysis**

The requirements were gathered through a variety of methods, including interviews with various hospital workers and research. As a result of the responses, the functional and non-functional requirements were established. The primary topic raised in the comments was the need to divide functionalities between users. As a result, the system must be designed in such a way that a user can only access certain functions and features. Furthermore, the system should automate the generation of reports and streamline the retrieval of records.

### **3.4 User Identification**

The hospitals in Ghana with special attention towards the low resource hospitals are the intended users of this hospital management system. Because the system is offered as a web application and a mobile application, users must have access to a computer or a smart phone with an internet connection to utilize it. Furthermore, users should have a basic familiarity of computers and the internet. They should also have a teachable spirit in case users are educated through workshops.

The user role includes the administrator, receptionist, doctor, nurse, pharmacists, and lab technician.

### **3.5 System Overview**

The functional and non-functional requirements of the system are as follows:

#### **3.5.1 Functional Requirement of the Hospital Management System**

##### **General Functional Requirements**

1. Adding Patients: This hospital management system will enable staff to include new patients into the system.

2. Assigning IDs to the patients: This hospital management system will enable the staff specifically the receptionist to provide a unique ID to each patient and then add them to the record sheet of the patient. This will enable the patient to utilize the ID throughout his or her stay in the hospital.
3. Mandatory Patient Information
4. Updating information of the patient: This hospital management system will enable users to update the information of the patient when deemed necessary.

The following are the functional requirements of specific users of the system:

Table 3.1: Functional Requirements of the various users of the system

Administrator	Receptionist	Nurse	Doctor	Lab Technician	Pharmacist
Create, read, update, and delete users	Activate and Deactivate the Patient Folder upon Visit	Create, read, update, and delete vitals	Create, read, update, and delete prescriptions	Add Test Type	Create, read, update, and delete drugs
View Patient Data Request from other hospitals	Create, read, update, and delete bills	Create, read, update, and delete allergies	Create, read, update, and delete records	Add Lab Test in relation to a particular visit	View all the patients that visited the hospital on that day
Create, read, update, and delete the specialization form	Set appointments between doctors and patients/visitors	View all the patients that visited the hospital on that day	Edit schedule	View all the patients that visited the hospital on that day	Confirm that patients have picked up their prescriptions
Assign specializations to doctors	View all the patients that visited the hospital on that day	Create, read, update, and delete vaccines	Assign vaccines and allergies to patients		
Create, read, update, and delete pharmacy	Create, read, update, and delete patients	Create, read, update, and delete Ghana Card	View drugs in the system and assign them to their respective prescriptions		
Create, read, update, and delete vaccine	Create, read, update, and delete insurance		View all the patients that visited the hospital on that day		
Create, read, update, and delete insurance	Create, read, update, and delete visits				
View patient information and records					



### 3.5.2 Non-Functional Requirement of the Hospital Management System

1. Login ID: Any user who needs to get access into the system needs a Login ID and a password for security reasons.
2. Response Time: The system provides acknowledgement in a few seconds once the patient's information is checked.
3. Availability: The system will be available all the time.
4. Maintainability: The system can be restored and upgraded easily.
5. Adaptability: The system will be able to adjust to new conditions.
6. Flexibility: The system can be easily modified.
7. The database of the support should be able to support large amounts of data.

### 3.6 Component Selection

This section focuses on the procedures used to pick the final components that will be employed to make the project a success.

#### 3.6.1 Pugh Matrix

Table 3.2: Pugh Matrix Legend

KEY	
0	Equal to the baseline
+	Better than the baseline
-	Worse than the baseline

## Pugh Matrix for Microcontroller

Table 3.3: Pugh Matrix for the Microcontroller

Criteria	Baseline	Weight	Alternatives	
	NodeMCU ESP8266		Arduino Uno	Esp32
Ports	0	5	-	+
Availability	0	3	+	+
Cost	0	2	+	+
Total			5	10

### 3.6.2 Final Components

#### Hardware:

Table 3.4: Hardware components

Component	Function
Esp32	It's a series of low-cost, low-power system on a chip microcontroller with Wi-Fi and dual-mode Bluetooth built in. The Esp32 was utilized in this project to assist in the transfer of data from the pulse oximeter to the management system's database via the Bluetooth module.
OLED Display	They are electronic visual panels that harness organic light-emitting diodes for their core illumination power. They present images, text, videos and more on the screen. The OLED Display will display the oxygen level and the percentage level on the screen.
NPN Transistors – BC547	These transistors are normally used as amplifiers when power is applied to the base. It flows from the collector to the emitter.

Component	Function
Photo Diode	They are semi-conductor devices with a P-N junction that converts photons into electrical current. It is used for the accurate measurement of light intensity.
IR Sensor	It is used to detect specific properties in its environment. It accomplishes this by producing or sensing infrared radiation.

**Software:**

Table 3.5: Software components

Software	Role
Django	Django is a high-level Python web framework for building secure and maintainable websites quickly. It is very efficient. This will be utilized for the project's backend.
React	React is an open-source toolkit for creating single-page apps with a focus on user interfaces. It enables the creation of reusable user interface components. It is responsible for handling the view layer in online and mobile apps. It will be utilized for the project's frontend.
MySQL	MySQL is a database management system that will be used to store patient information in healthcare facilities.
Bootstrap	Bootstrap is a powerful front-end framework for building modern websites and online apps that look great on any screen. As a result, our users will be able to view our management system on any device of their choice thanks to this framework.

**3.7 Software System Design**

This section explains how the software aspect of the project is broken down. That is, to make it easier to save patient information, receive data from the pulse oximeter, securely transfer patient

data from one hospital to another upon request, and allow patients to access their personal information whenever they want.

### **3.7.1 Hospital Database**

The Hospital Database's only objective is to store critical information, not just about patients, but also about the hospital, to increase its efficiency. The hospital must maintain the following items as part of the project's scope: user information, appointment times, scheduling times, patient vitals, and reports.

A patient is enrolled into the system during their initial visit to the hospital, which generates a document containing basic information on the individual, and new documents are uploaded at subsequent check-ups. The Django rest framework database is used to retain a patient's data in their own collections.

Patients will also be able to fill out their patient forms using QR codes. This is aimed towards decreasing the load on receptionists in low-resource hospitals when there is a long line and reducing the stress that comes with it.

### **3.7.2 Pulse Oximeter Data**

The aim of this sub-section is to highlight the process on how the data from the pulse oximeter will be transferred into the database.

With this side of the project, the Esp32 would be made to interact indirectly with the database server of the website via HTTP/HTTPS.

The following are the steps that would be performed to help make this project a success:

1. The Esp32 would make an HTTP request to the Web server
2. The Web server would run the PHP script
3. The PHP script would get data from the HTTP request, process the data, and then interact with the database of the Hospital Management System.
4. The PHP script would process the result and return the results to the Esp32 via the HTTP response.

### **3.7.3 Transfer Patient Data**

The transfer patient data section elaborates on the process by which the data of a patient, specifically the record of a patient is transferred between hospitals. This part of the project will be accomplished using JSON Web Token. The JSON Web Token (JWT) is an open standard that offers a compact and self-contained method for securely communicating information as a JSON object between parties. The JWTs are used as a secure way to authenticate users and share information.

Hospitals will be allowed to obtain data from other hospitals on their patients. Through a form, the user who is interested in the patient's information will request a token from the other facility. From there, a link will be provided to the user's email address after validation, however it will only be valid for a limited time. The tokens are only allowed to be used once before being stopped by the backend. The patient's data will be stored in a JSON format, which the user will need to download to access it.

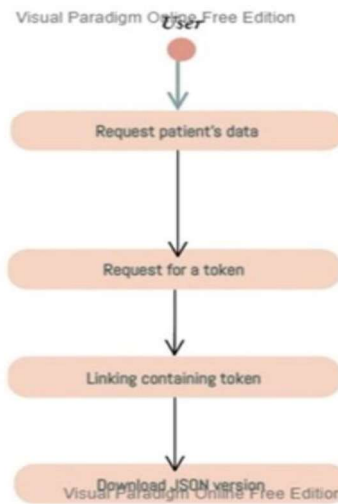


Figure 3.1: A use case diagram for the transfer process

### 3.8 Hardware System Design

The finalized components for the Hardware System Design for this project which is the Pulse Oximeter include the following:

1. Esp32 microcontroller
2. OLED Display
3. Infra-Red Led
4. Photo Diode
5. Red Led
6. NPN Transistors (BC547)
7. Resistors (4.7k & 10k)

#### 3.8.1 The Pulse Oximeter

The purpose of the pulse oximeter to assess patients' oxygen levels and then communicate the information to the patient's database.

The pulse oximeter employing EasyEDA is schematically depicted in the diagram below. Esp32, OLED Display, Infrared led, photo diode, red led, NPN transistors (2), and resistors (6) make up this gadget.

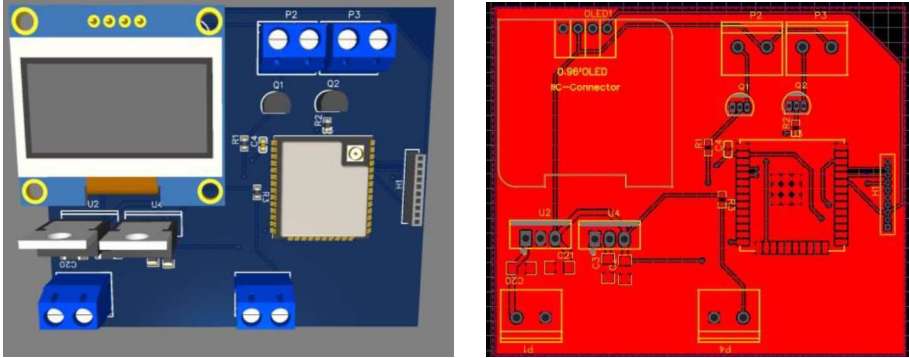


Figure 3.2: Simulation of the circuitry

## **CHAPTER 4: Implementation**

### **4.1 Hospital Management System Implementation**

This chapter describes the steps that went into implementing the Hospital Management System phase of the project.

It comprises of:

- i. Hospital Database
- ii. Transfer Patient Data
- iii. Pulse Oximeter Data

#### **4.1.1 Hospital Database**

The project commenced with a visual representation of how the project would look like i.e., an Entity Relationship (ER) Diagram to aid in the development of the project. An entity relationship diagram is a graphical representation that depicts relationships among people, objects, places, concepts, or events within an information technology system. It gave a snapshot of how the entities would relate to each other hence, it can be called a blueprint that underpins a project.



The diagram below is the ER Diagram for the Holified Hospital Management System:

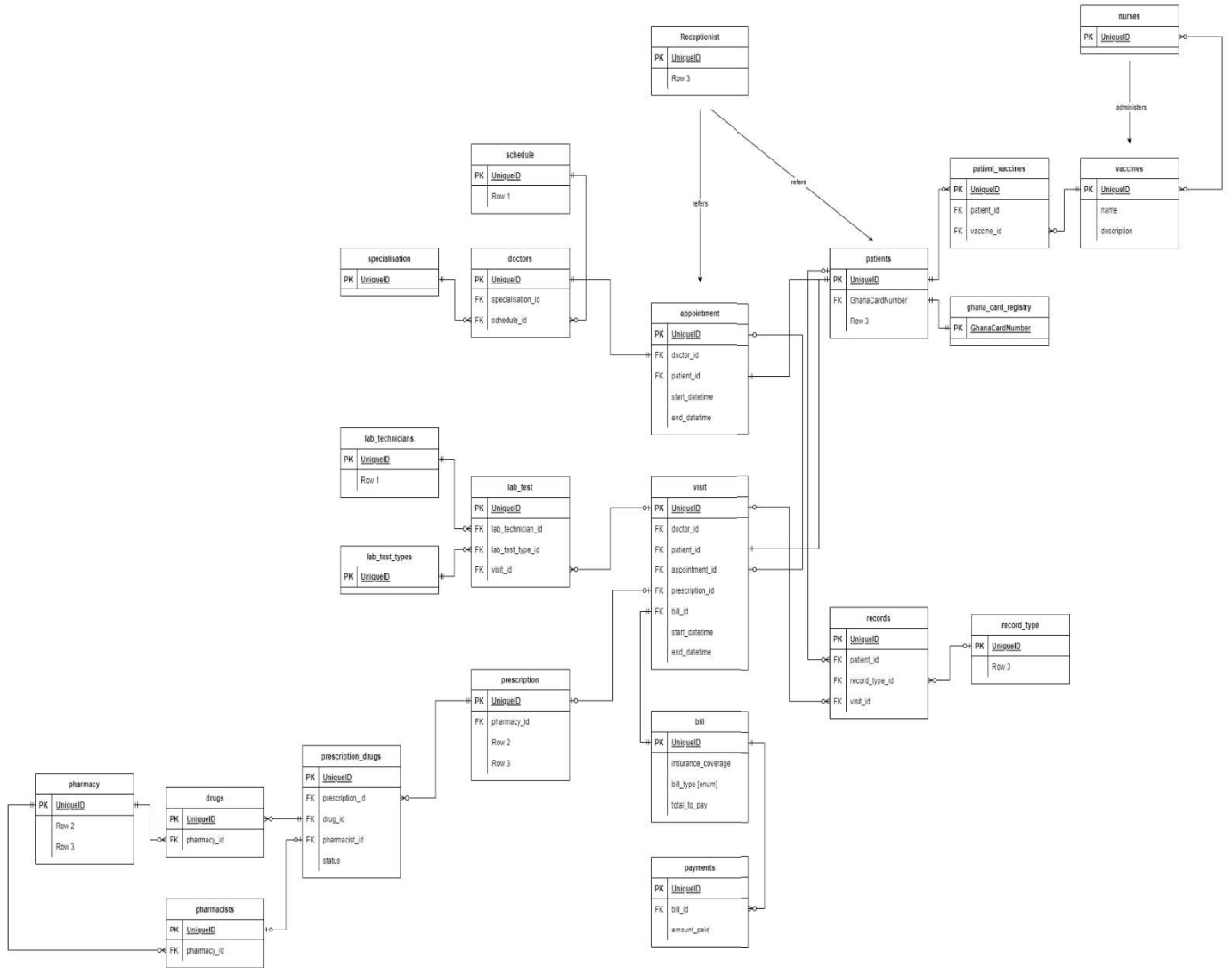





Figure 4.1: The ER Diagram for the Hospital Management System

Table 4.1: Legend for the ER diagram

ER Symbol	Definition
	Optional to many relationship
	One to mandatory relationship
	One to optional relationship

## Case Study

This project's ER diagram focuses on the comprehensive treatment of a patient from the moment of entry to the point of exit. Upon arrival, the patient is greeted cheerfully by the receptionist, who, depending on the patient's hospital status, either generates or updates the patient's file. The patient goes to the nurse to have his or her vitals taken. To add on to the uniqueness of this system, a smart device named the pulse oximeter will be used to take the SpO2 value of the patient and then transferred directly into the database of the hospital's management system. However, the other vitals will be measured manually. The patient is then referred to a doctor to have his or her concerns handled. If the patient requires testing, he or she will be referred to a lab technician who will perform the required steps. The doctor concludes about the patient's problem, then directs him/her to the pharmacist, who will assist him/her in selecting the prescribed medications for treatment.

Patients and others will be able to arrange appointments with doctors as well as view the times they are available for booking to avoid scheduling conflicts, using this management system.

The interface includes forms, tables, inputs, and dashboards. It is important to note that the system was developed for various users including pharmacists, lab technicians, receptionists, doctors,

administrators, and nurses. Therefore, all users have the same theme for their interface. The user interface is mobile friendly, in the sense that it is highly responsive to screen changes and provides a sidebar which allows users to navigate through the system on any device easily.

### **The Management System**

All users would have to log in to gain access to the system. The management system opens with a landing page which has two buttons namely the create appointment and login. Since patients are not recognized users in the system but need to gain access to a part of the system, they have been provided with the option of booking an appointment with the doctor on the landing page. The login button allows the various users on the system to gain access into the Holidified management system. To gain access into the system, the user would have to select the specific hospital, type in the user id or the Ghana card personal id number and the password. This authentication process was thoroughly thought of to enhance the safety of the management system. Aside that, it also allows the system to track the actions and behaviors of the users registered into the account.

The validate function in the Login Authentication Code in Figure 4.2 crosschecks the `employee_id` and password entered into the system before giving access into the user. Depending on the correctness or incorrectness of the details and what action takes place afterwards.

```
class LoginSerializer(serializers.Serializer):
    employee_id = serializers.CharField(write_only=True)
    password = serializers.CharField(max_length=128, write_only=True)
    access = serializers.CharField(read_only=True)
    refresh = serializers.CharField(read_only=True)
    user = UserSerializer(read_only=True)

    def validate(self, data):
        employee_id = data['employee_id']
        password = data['password']
        user = authenticate(employee_id=employee_id, password=password)

        if user is None:
            raise serializers.ValidationError("Invalid login credentials")

        if not user.is_active:
            raise serializers.ValidationError("user account has been deactivated")

        try:
            refresh = RefreshToken.for_user(user)

            return {
                'refresh': str(refresh),
                'access': str(refresh.access_token),
                'user': user
            }
        except User.DoesNotExist:
            raise serializers.ValidationError("Invalid login credentials")
```

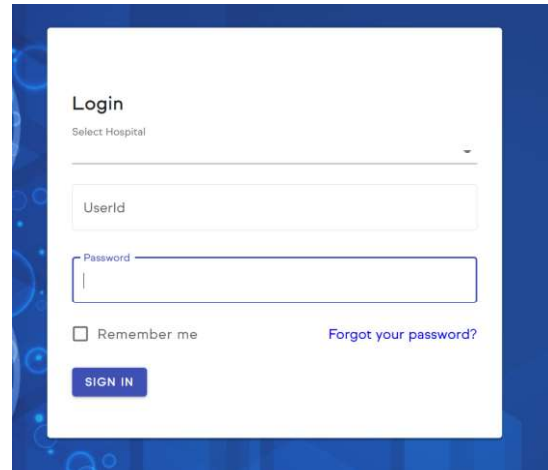


Figure 4.2: Login Authentication code & the Login Page

## The various users of the system

There are six users of the Holified management system. Each user has a different interface pertaining to what they do.

### 1. Administrator

#### User Registration

The administrator oversees the registration of users in the system. A user will be able to sign in into their respective accounts using their employee id and a password. In registering the nurse, doctor, lab technician, pharmacist and receptionist, the administrator only requires the employee id, the role, and the password. The default password of every new user is “password” whereas the employee id can be generated by the administrator or be auto generated. Figure 4.4 demonstrates how it is done.

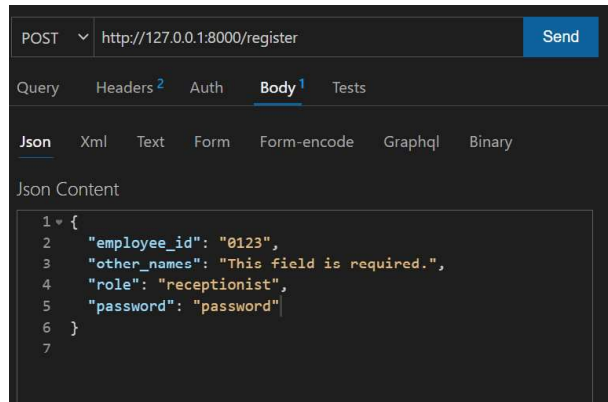


Figure 4.3: Receptionist Registration

## **Dashboard**

The dashboard is the information management tool that shows relevant information that cuts across to all the respective users. It contains a side bar, the total appointment per day, total patients per day and the average number of patients per day. It also shows the patient list, the appointment list, the doctor of the month and the appointment status pie chart. It is very versatile.

In generating the data for the total appointment per day, total patients per day, the average number of patients per day and the appointment status pie chart a function called aggregate was used. Aggregate in Django helps in calculating the values for the entire query set.

The function in Figure 4.4 takes two query parameters that is operation (op) for the sum, count, maximum, average, and minimum and grouping which differs depending on the field the user chooses. Also, the aggregate function can be called in any of the models required of it. For the dashboard specifically, it was used in the appointment and patient model due to the results needed to be seen at the frontend.

```

6  op = {
7    'sum':Sum, 'max':Max,
8    'min':Min, 'avg':Avg,
9    'count':Count,
10 }
11
12 class Aggregation(object):
13
14     @action(detail=False, methods=['get'], url_path='aggregate')
15     def aggr(self, request):
16         params = dict(request.query_params)
17
18         func, group = op.get(params.pop('op', None)[0], None), params.pop('group', None)
19
20         if not func or not group:
21             raise ValidationError('no aggregation function provided or group missing')
22
23         try:
24             model = self.serializer_class.Meta.model
25             # res = self.queryset.filter(**params).aggregate(*[func(group) for group in group])
26             res = model.objects.values(*group).filter(**params).annotate(**{field+'_'+func.__name__.lower(): func(
27                 return Response(res)
28         except Exception as e:
29             raise ValidationError(e)
30

```

Figure 4.4: Aggregate function

In generating data from the backend to the frontend, the required API ENDPOINT had to be called.

```

componentDidMount() {
  if (this.state.userRole === Roles.ADMIN) {
    sendGetRequest(API_ENDPOINTS.PATIENT).then(getResponse => {
      return this.setState({
        fetchedData: getResponse.data.results,
        tableColumns: this.PatientColumns,
        title: 'Patient Data',
      });
    });
  });
}

```

Figure 4.5: The componentDidMount function

The componentDidMount function helps in giving functional components the ability to manage states.

Aside calling the required API ENDPOINTS, the columns in the table ought to match those at the backend, otherwise, they will not be seen at the front end. An action table was created to aid in accomplishing the task of the dashboard. That is, being able to view the vitals, record, lab test and pharmacist form.

```

PatientColumns = [
  { title: 'First Name', field: 'first_name' },
  { title: 'Last Name', field: 'last_name' },
  { title: 'Other Name', field: 'other_names' },
  { title: 'Sex', field: 'sex' },
  { title: 'Ghana Card Number', field: 'ghana_card_number' },
];

actions = [
  {
    icon: Create,
    tooltip: 'vitals',
    onClick: (event, rowData) => {
      this.handleClickOpen(rowData);
    },
  },
  {
    icon: LocalHospital,
    tooltip: 'doctor',
    onClick: (event, rowData) => {
      this.handleClickOpen(rowData);
    },
  },
  {
    icon: Visibility,
    tooltip: 'lab technician',
    onClick: (event, rowData) => {
      this.handleClickOpen(rowData);
    },
  },
  {
    icon: LocalPharmacy,
    tooltip: 'pharmacy',
    onClick: (event, rowData) => {
      this.handleClickOpen(rowData);
    },
  },
];

```

Figure 4.6: The columns in the patient data table and the action table

```

render() {
  return (
    <PageContainer heading={IntlMessages id="pages.dashboard"} breadcrumbs={this.breadcrumbs}>
      <GridContainer>
        <Grid item xs={12}>
          <Trends />
        </Grid>
        <Grid item xs={8}>
          <PatientDataTable
            tableColumns={this.state.tableColumns}
            tableData={this.state.fetchedData}
            tableTitle={this.state.title}
            Action={this.actions}
          />
        </Grid>
        <Grid item xs={4}>
          <StatusPieChart />
        </Grid>
      </GridContainer>
    </PageContainer>
  );
}

```

Figure 4.7: The patient data table

The patient data table shows how the table should be laid out on the page. It also includes the prop function, which is used to transmit data from the backend table to the frontend table.

### **Patient Data Request**

Every hospital in the system has a form that allows them to obtain patient data over a period. The administrator must fill out this form with the required patient's name, the email address to which the patient's data will be sent, the administrator's phone number, the type of record, which is either a lab record or a visit record, and the date range of the patient's required records.

In creating the patient data request form, which applies to the other forms that were created, a patient data request form.js file was first created. It contained various functions that included the column fields, the form validation which had specific requirements for each field, a display alert which displayed whether the form was submitted successfully or there was an error and goes further in detailing what the error was, and how the form should appear at the frontend of the page.

```

const INITIAL_FORM_STATE = {
  name: '',
  email: '',
  phone: '',
  ghana_card_number: '',
  start_date: '',
  end_date: '',
  recordtype: ''
};

const recordtypeOptions = {
  visit: 'visit',
  lab: 'lab'
};

const FORM_VALIDATION = Yup.object().shape({
  name: Yup.string()
    .required('Please enter a first name.')
    .min(2, 'First name should be at least 2 characters'),
  email: Yup.string(),
  phone: Yup.number()
    .integer()
    .typeError('Please enter a valid phone number')
    .required('Please enter a phone number')
    .min(10, 'Phone number should be at least 10 digits for local numbers'),
  recordtype: Yup.string().required('Please indicate your recordtype.'),
  start_date: Yup.date()
    .required('Please enter the start date'),
  end_date: Yup.date()
    .required('Please enter the end date'),
});

const displayAlert = (alertMessageType = 0, errorState = null) => {
  if (alertMessageType === 1) {
    return <SuccessAlert message='New patient data requested successfully.' />;
  }
  if (alertMessageType === -1) {
    return (
      <ErrorAlert
        message='Sorry, patient data was not added for the following reasons: '.concat(JSON.stringify(errorState))
      />
    );
  }
};

```

Figure 4.8: The initial form state of the patient data request form & the form validation and display alert function of the patient data request form

A Patient Data Request folder was created with its index.js file that imported the patient data request form. The index.js file of the patient data had functions that elaborated on the page layout of the form.

```

const PatientData = () => {
  return (
    <PageContainer heading={<IntlMessages id="pages.patientdata" />} breadcrumbs={breadcrumbs}>
      <GridContainer>
        <Grid item xs={12}>
          <PatientDataForm />
        </Grid>
      </GridContainer>
    </PageContainer>
  );
};

```

Figure 4.9: The patient data function



A route was created in the route index.js file to connect the backend of the patient data request file to the frontend for data exchange.

```
return (  
  <React.Fragment>  
    <Switch>  
      <RestrictedRoute path="/dashboard" component={Dashboard} />  
      <RestrictedRoute path="/patients" component={Patient} />  
      <RestrictedRoute path="/patient-form" component={PatientForm} />  
      <RestrictedRoute path="/visits" component={Visits} />  
      <RestrictedRoute path="/appointments" component={Appointments} />  
      <RestrictedRoute path="/vaccine" component={Vaccine} />  
      <RestrictedRoute path="/ghanacard" component={GhanaCard} />  
      <RestrictedRoute path="/specialisation" component={Specialisation} />  
      <RestrictedRoute path="/user" component={User} />  
      <RestrictedRoute path="/patientdata" component={PatientData} />  
      <RestrictedRoute path="/expirydate" component={ExpiryDate} />  
      <RestrictedRoute path="/allergies" component={Allergies} />  
      <RestrictedRoute path="/bill" component={Bill} />  
      <RestrictedRoute path="/labtest" component={LabTest} />  
      <RestrictedRoute path="/prescription" component={Prescription} />  
      <RestrictedRoute path="/records" component={Record} />  
      <RestrictedRoute path="/insurance" component={Insurance} />  
      <RestrictedRoute path="/testtype" component={TestType} />  
      <Route path="/signin" component={Login} />  
      <Route path="/forgot-password" component={ForgotPasswordPage} />  
      <Route component={Error404} />  
    </Switch>  
  </React.Fragment>  
)
```

Figure 4.10: The route of the patient data request form and the other forms

The screenshot shows a web form titled "The HOLIFIED Patient Data Request Form". It contains several input fields: "Name", "Email", "Phone", "Record Type" (a dropdown menu), "Start Date" (with a date picker icon and the format "mm/dd/yyyy"), and "End Date" (with a date picker icon and the format "mm/dd/yyyy"). At the bottom of the form, there are two buttons: a blue "SUBMIT FORM" button and a red "CANCEL" button.

Figure 4.11: The HOLIFIED Patient Data Request Form

## Specialization

The hospital administrator is the only user admitted registering the other users (receptionist, nurse, doctor, lab technician, pharmacist) unto the system. It follows the same procedure in the creation of a form, like the patient data request form. However, the administrator has an additional role of assigning the specializations to the various doctors and pharmacist. In registering the doctor and

the pharmacist, the specialization and pharmacy forms need to have data for the various users to be assigned to them respectively.

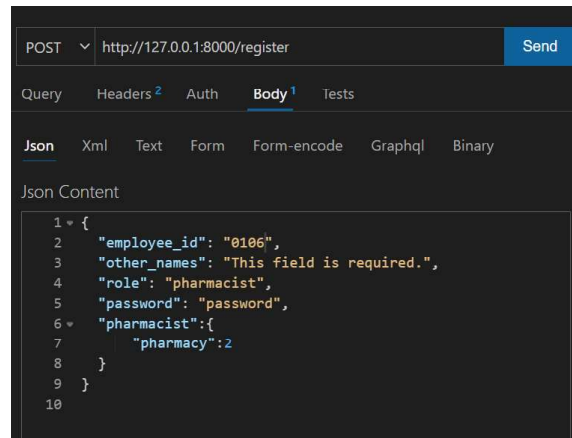


Figure 4.12: Pharmacist Registration

## 2. Receptionist

### Registration and updating patient information

The receptionist oversees the registration of new patients and updating the files of registered patients. As stated earlier, the creation of forms follows the same procedure as the patient data request. However, it differs in updating the files of already existing patients. Updating the files of the registered patients allows the receptionist and any other user assigned to that role to create, read, update, and delete (crud) the patient form.

The crud process was used by all the other users however, the endpoints differed. To begin with, prop functions had to be created in the Patient Data Table, which is a table that hubs all the data pertaining to any user or form. The prop function helps pass data from one component to another, thus offering a channel through which the components can communicate. Hence the prop function

was created in the Patient Data Table to serve as a channel between the patient form index.js file and the patient data table.

```
<MaterialTable
  className={classes.root}
  icons={tableIcons}
  title={props.tableTitle}
  columns={props.tableColumns}
  data={props.tableData}
  actions={props.Action}
```

Figure 4.13: The prop functions

For a user to be able to create, read, update, and delete a page, the following functions were required:

1. Columns – The field details
2. Title – The title of the page
3. Data – Data from the backend
4. Action – An action statement addressing what the user would be able to do

Prop functions were created for the title, columns, data, and action. Functions were created to define the columns, title, data, and action in the patient index.js file. The fields in the column had to match the fields in the backend, otherwise, the data could not be displayed.

A useEffect function was created which aids in requesting for data from the endpoint and maps the response to the setFetchedData. The patient data table was imported to the patient form which allows the assigned user to view the list of patients stored at the backend as well as perform the set actions required of the user. For the receptionist, the user has the ability of deleting, editing, and creating a new patient.

```

const columns =[
  { title: 'ID', field: 'id' },
  { title: 'First Name', field: 'first_name' },
  { title: 'Last Name', field: 'last_name' },
  { title: 'Other Names', field: 'other_names' },
  { title: 'Date Of Birth', field: 'date_of_birth' },
  { title: 'Sex', field: 'sex' },
  { title: 'Height', field: 'height' },
  { title: 'Ghana Card Number', field: 'ghana_card_number' },
  { title: 'Nationality', field: 'nationality' },
  { title: 'Code', field: 'code' },
]

```

Figure 4.14: The column function

```

const Patient = () => {
  const title = "Patientlist";
  const [fetchData, setFetchData] = useState([]);

  useEffect(() => {sendGetRequest(API_ENDPOINTS.PATIENT).then(getResponse => {
    return setFetchData(getResponse.data.results)
  })}, [sendGetRequest, setFetchData])

  const openPage={() => {
    window.location.assign("/patient-form");
  }}

  return (
    <PageContainer heading={<IntlMessages id="pages.patients" />} breadcrumbs={breadcrumbs}>
      <GridContainer>
        <Grid item xs={12}>
          <button type="Add New Patient" color="primary" onClick={() => openPage()}>Add New Patient</button>
          <PatientDataTable tableColumns={columns} tableData={fetchData} tableTitle={title} />
        </Grid>
      </GridContainer>
    </PageContainer>
  );
};

```

Figure 4.15: The patient and useEffect function

The same process of crud was applied to the other forms index.js file of the system.

## Appointment

The appointment form allows patients and visitors of the hospital to book the services of any of the health practitioners available depending on the booking times. It indicates the days and times the respective doctors are free and produces feedback as to whether the appointment has been honored, cancelled, rescheduled, confirmed, and postponed.

The screenshot shows a web form titled "The HOLIFIED Appointment Form". It contains the following fields and controls:

- "Select Doctor": A dropdown menu.
- "Date of Appointment": A date input field with the placeholder "mm/dd/yyyy" and a calendar icon.
- "Name": A text input field.
- "Start Time": A time input field with the placeholder "--:--" and a clear button.
- "End Time": A time input field with the placeholder "--:--" and a clear button.
- "Phone": A text input field.
- "Status": A dropdown menu.
- "SUBMIT FORM": A blue button.
- "CANCEL": A red button.

Figure 4.16: The Appointment Form

### **Flagging Patient Visit**

In creating the flag patient visit file, a js file was created which served as a config file and is expected to update every time the receptionist wants to turn on or off the feature.

In creating the file, it requires a unique name that will later be called, description that describes the functionality and an active flag that determines whether the toggle is on or on are required. The flags are stored in an array.

However, a function was added to an app.js and called in the required component file to store the flags in a local storage. Since localStorage only handles strings, the JSON. stringify () function converts JavaScript objects to strings.

A new react component was constructed to access these feature flags in React and show/hide features based on these feature flags.

Two props were added by the component:

1. A name for the flag to be compared to

## 2. A child content to be used

The feature is retrieved from localStorage and tested to see if it is activated. The feature is rendered if it is active; else, null is returned.

Table 4.2: Flagging Patient Visit Table

Name	1 <sup>st</sup> Doctor	2 <sup>nd</sup> Doctor
Traci Gyebi	True	False
Lovelace Gyebi	False	True

In implementing this requirement, a receptionist would have to ensure whenever a patient visits the doctor, the patients file is left on flag, and is unflagged after leaving the hospital. The table above gives a comprehensive explanation of flagging a patients visit.

## 3. Nurse

The nurse aside filling up the necessary forms pertaining to the patients, has a unique requirement of storing the vitals as a json field and has a pop-up menu displayed whenever the SpO<sub>2</sub> value is entered into the system through the Esp32 of the circuitry.

In storing the vitals as a json field, the data was stored in a set of key-value pair. It follows the same process in registering a new user into the system by the administrator.

### **SpO<sub>2</sub> values and the pop-up message**

The SpO<sub>2</sub> values were generated from the pulse oximeter and connected to the database through the Esp32. Whenever the patient's oxygen saturation value was recorded between 95% and 100%,

a pop-up message appeared on the screen saying, “Awesome, you’re set to go” while if the patient’s oxygen saturation value fell below 95%, a pop-up menu appears saying, “Hey there! Take care of yourself.”

A file under component was designated for the pop-up message named, “popup.js”. The pop-up component was managed using the state variable. Since the state variable was used, a toggle method had to be created to manage the show or hide of the pop-up component. The toggle method was passed inside the popup.js file in component.

#### **4. Doctor**

##### **Schedule**

The schedule form makes use of the function in Figure 4.17. It allows the doctors to input their respective schedules for an appointment form to be generated for the patients and visitors interested in booking an appointment to see the various medical practitioners. The medical practitioners can choose the time they would like to start and end their appointments, the durations, and the dates. From the schedule form that will be filled by the medical practitioners available, the visitors and the patients have the ability of selecting their preferred doctor.

While the patient or visitor is filling the form, a patch and put request is pushed to the backend to help fill the option form depending on the doctors available.

The epoch function was the action responsible for allowing the doctors to choose their available days and times.

```
@action(detail=False, methods=['get'], url_path='available-epochs')
def get_free_epochs(self, request):
    params = request.query_params
    start, end, duration, doctor, specialisation, epochs = params.get('start', ''), params.get('end', ''), \
        params.get('duration', 30), params.get('doctor', None), params.get('specialisation', None), {}

    if start==' ' or end==' ':raise ValidationError("start and end required", code=400)

    try: dt_range, duration = date_range(start, end), int(duration)
    except Exception as e: raise ValidationError(e, code=400)

    doctors = User.objects.all().filter(role=User.Types.DOCTOR)

    if specialisation: doctors = doctors.filter(doctor_specialisation__exact=specialisation)
    if doctor: doctors = doctors.filter(pk__exact=doctor)

    for date in dt_range:
        epochs.update({'[date]':{}})
        for doctor in doctors:
            schedule = Schedule.objects.all().filter(doctor=doctor, day=date.weekday()).first()
            appointments = Appointment.objects.all().filter(doctor=doctor, date__exact=date, status__in=[

                if not schedule:continue

            ex_epochs = list(appointments.values()) + list(schedule.breaks.all().values())
            ex_epochs = [(combine_dt_time(date, epoch["start_time"]), combine_dt_time(date, epoch["end_time"])]
            start_time, end_time = combine_dt_time(date, schedule.start_time), combine_dt_time(date, sche
```

Figure 4.17: The epoch function

### Assigning drugs to the prescriptions

Similar to how the administrators assigned the doctors to their respective specializations, the doctors have the ability of assigning the drugs available to the patient’s prescription before pushing it to the pharmacist.

## 5. Lab Technician

The lab technician interface was built to create, read, update, and delete (crud) the various test types and the lab test pertaining to a particular visit. This follows the same procedure as the other forms discussed above.

## 6. Pharmacist

The pharmacy interface was built to receive information from the doctor concerning the patient, view patients’ data, confirm the pickup of drugs.



### 4.1.2 Transfer Patient Data

To aid in the transfer of patient data, JSON Web Token Authentication will be used by the client/server applications.

JWT is an authorization token that should be included in all requests. It is acquired by exchanging a user detail for an access token. The access token lasts for a short period of time because of its security feature.

The JWT is divided into three parts that is: the header, the payload, and the signature. The information was encoded using Base64. The signature is also issued by the JWT backend, using the header base64 + payload base64 + secret key. Anytime a request is meted out, the signature is verified. If by any chance, information in the header or in the payload was changed by the client, it will invalidate the signature. The accurate and only way of checking and validating the signature is by using the application's secret key.

To begin with, the `django-rest-framework-simple-jwt` was installed onto the computer. The Django Rest Framework (DRF) web interface was used to access the API endpoints. The next step was to authenticate and obtain the token. The endpoint was `/api/token` and it only accepted POST requests.

After an administrator has sent a request for a patient's record through the patient data request form, the administrator of the said patient's hospital pushes a JSON token to the administrator via the email that was asked for in the patient data request form. However, in the project, mailtrap was used to avoid creating other domain accounts email. Mailtrap is a service for the safe testing of emails sent from the development and staging environments [20]. It catches emails in a virtual inbox so that one can test and optimize their email campaigns before sending them to real users.

The response body generated two tokens; however, this project is only interested in the access token. The access token was stored on the client side in the local storage.

To access the protected views on the backend (i.e., the API endpoints that require authentication), one is expected to include the access token in the header of all requests.

The access token was eligible for only thirty minutes.

After thirty minutes, the token will expire and to gain access to view the patient medical record again, one would have to request for another token.

```
@action(detail=False, methods=['get'], url_path='transfer')
def transfer(self, request):
    auth = request.query_params.get('auth', None)

    if not auth:
        raise ValidationError('missing token')

    data = jwt.decode(auth, SECRET_KEY, algorithms="HS256")

    if data['expires']:
        try:
            expires = datetime.strptime(data['expires'], '%Y-%m-%d')
        except Exception as e:
            raise ValidationError('something went wrong, token data format mismatch')

        if not expires > datetime.now():
            raise ValidationError('token expired')

    try:
        req = RecordRequest.objects.get(id=data['id'])
        records = self.queryset.filter(created_date__range=[req.start_date, req.end_date]).all()

        serializer = self.serializer_class(records, many=True, context={'request':request})

        res = serializer.data
        page = self.paginate_queryset(res)
        if page is not None:
            res = self.get_paginated_response(page).data
        return Response(res, status=200)
```

Figure 4.18: Transfer function

The transfer function in Figure 4.18 above validates the patient data request that is submitted to the administrator of a hospital. It ensures the date range requested from the administrator has the patient's records recorded, likewise it sets the time for the token to expire.

### **4.1.3 Pulse Oximeter**

#### **4.1.3.1 How the pulse oximeter operates**

The percentage oxygen saturation in the blood is calculated by the amount of different light absorbed by the pulse oximeter. The oximeter is made up of a photodiode, a Red LED, and an infrared LED, with both frequencies turning on to pulse at the same frequency.

A fingertip is put between the photodiode and the LEDs to correctly capture the measurement. Blood that is deoxygenated absorbs more red light and passes through more infrared light, whereas blood that is oxygenated absorbs more infrared light and passes through less red light. At different time intervals, the processor estimates the ratio of red light received at the photodiode to infrared light.

#### **4.1.3.2 How the circuitry was built**

The OLED display's ground and VCC were connected to the Esp32's Vin and Ground. The OLED's SDA was connected to the Esp32's GPIO21, while the OLED's SCL was connected to the Esp32's GPIO22.

The photodiode was flattened to enable it to receive the more visible light from the source, as its spherical shape and dark color prevented it from receiving more visible light in the first place. For the same reason as the photodiode, the two LEDs, namely the red and infrared LEDs, were glued together.

The resistor connected to the red LED, which limits the flow of current from the Esp32, was likewise connected to GPIO12 of the Esp32, as did the resistor connected to the infrared LED. The photodiode's resistor was connected to the Esp32's GPIO33.

The transistor in the circuitry simplified the process of toggling between the infrared and the red LEDs. The two transistors were each connected to one of the LEDs' legs, with the other leg connected to the Esp32's ground.

The LEDs and photodiode calculated the functional oxygen saturation by measuring the transmission between the tissues.

The circuitry was powered by the Esp32, which provided 5 volts.

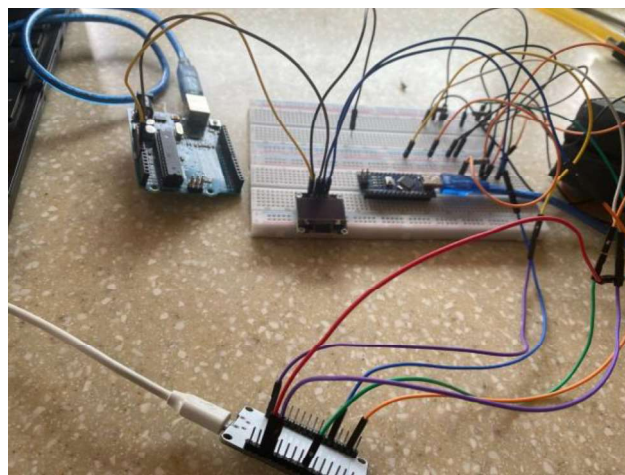


Figure 4.19: The circuitry of the pulse oximeter

#### 4.1.3.3 Programming and calculation of the pulse oximeter

The oxygen saturation level ( $SpO_2$ ) is a function of a parameter called R and is the fraction of oxygen-saturated hemoglobin compared to total hemoglobin.

$$R = \left( \frac{RED_{max} - RED_{min}}{RED_{min}} \right) / \left( \frac{IR_{max} - IR_{min}}{IR_{min}} \right) ;$$

Figure 4.20: The formula for finding R

Each device has its own R, and calibration is required to determine the function that connects R to  $SpO_2\%$ . To do so, the heartbeat's "period" (or how many milliseconds a beat lasts) was

measured and divided by the sampling rate to calculate how many samples make up a period. Since we read the IR led for 20 milliseconds and then the RED led for another 20 milliseconds, the sample rate is 40 milliseconds.

The time between two rising curves in the signal is known as the beat period.

The array's latest L samples (where  $L = \text{period} / 40$ ) were evaluated to determine the RED\_max, RED\_min, IR\_max, and IR\_min values.

The R value can be determined using the maximum and minimum values.

R, L, and period are calculated per beat; thus, the R calculus is done as well.

### **SATURATION IS A FUNCTION OF R (calibration)**

$$Y = k * x + m$$

Figure 4.21: How to calibrate the oximeter

To determine K and M, two points will be required (two pairs of SpO<sub>2</sub> and R values). Using another oximeter and reading the readings from its display is the only way to find these two locations.

Hence the new oximeter was used as a reference, reading the SpO<sub>2</sub> while measuring the R values with the pulse oximeter.

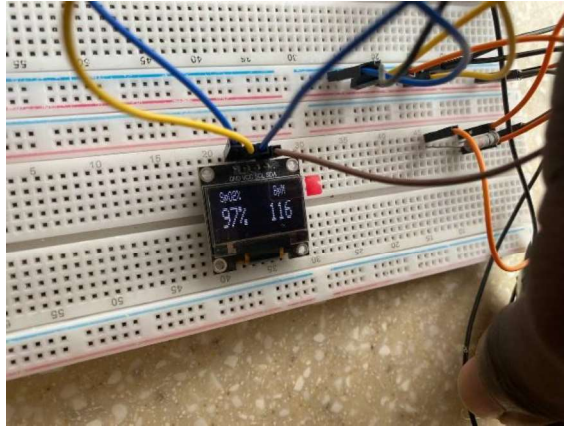


Figure 4.22: A reading from the pulse oximeter

#### 4.1.3.4 Transferring the Data unto the HMS' Database

The ESP32 was used because of its Wi-Fi module feature. Here, the data that is being read from the pulse oximeter is being transferred unto the database system of the hospital management system.

An HTTP Client code was written in Arduino to help push the code from the Esp32 to the database of the Holified management system.

In doing so, a network was set up to connect the Esp32 and the online database together.

```
wifiMulti.addAP("yawbrown", "yawbrown49");
```

Figure 4.23: A network was set up with the credentials above

The website's url had to be entered into the code to direct the path of the data from the Esp32 directly into the database of the Holified Management System.

```
USE_SERIAL.print("https://theholified.com/insert.php?patientID=0001&SpO2=97");
```

Figure 4.24: The website's URL

```
Writing at 0x0004c000... (84 %)
Writing at 0x00050000... (89 %)
Writing at 0x00054000... (94 %)
Writing at 0x00058000... (100 %)
Wrote 481440 bytes (299651 compressed) at 0x00010000 in 4.7 seconds
Hash of data verified.
Compressed 3072 bytes to 122...

Writing at 0x00008000... (100 %)
Wrote 3072 bytes (122 compressed) at 0x00008000 in 0.0 seconds (e
Hash of data verified.

Leaving...
Hard resetting...
```

Figure 4.25: The code uploaded successfully

The code was uploaded successfully and pushed unto the database of the Holified management system.

## **CHAPTER 5: Testing**

### **5.1 Overview**

This section of the project is important since it ensures that the system functions correctly and that the user's expectations are satisfied. As a result, this chapter's testing included the transfer of patient data, the hospital database, and ultimately the pulse oximeter.

### **5.2 Transfer of Patient Data Test**

The Transfer of Patient Data test was done to ensure that the system was working effectively and could be used to aid in the swift transfer of medical records from one hospital to another within the same management system. As stated earlier, the JSON Web Token was going to be used to aid in the swift transfer of medical records.

To test the transfer of patient data, a request was pushed through from one of the hospitals registered unto the management system to request for a token through the patient data request form. The administrator had the option of deciding whether or not to accept the offer. Upon acceptance, a link containing token was sent to the other hospital via email. Upon receipt, the JSON version was downloaded to retrieve the patient's medical records.

The token was only active for about 30 minutes after which it expired and could not be used to gain access into a patient's medical record.



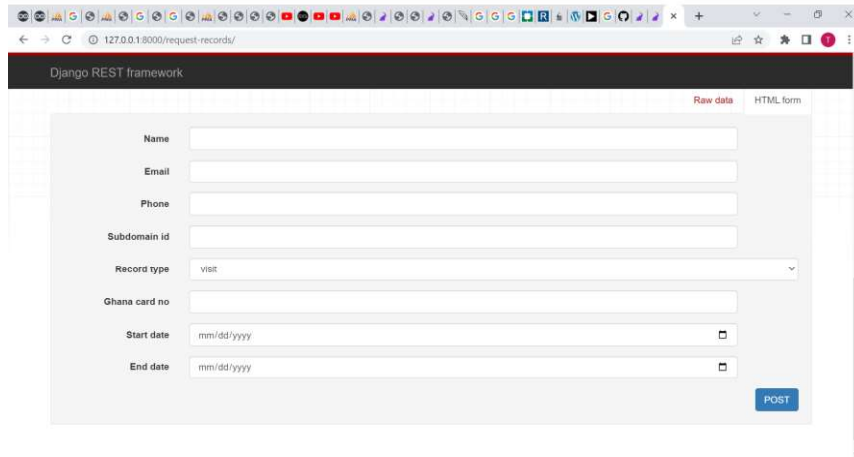


Figure 5.1 Filling the patient data request form

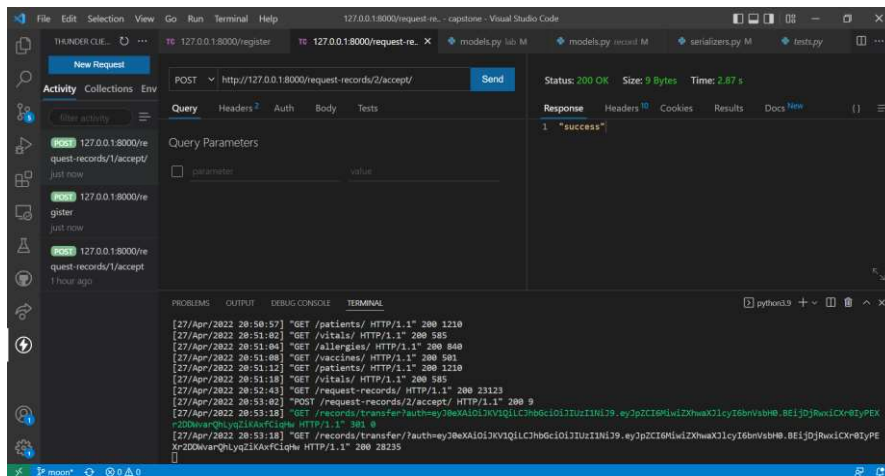


Figure 5.2: Accepting or Declining the Request

If the request had been declined, the url in figure 5.2 would have changed from accept to decline.



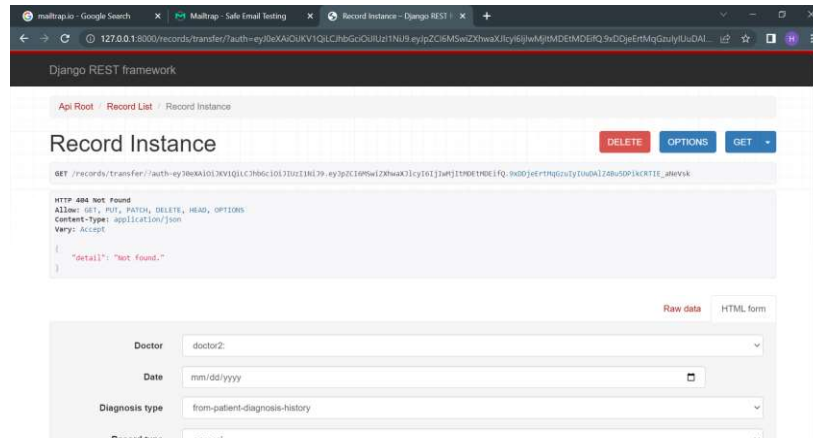


Figure 5.6: The user's record within an unavailable date range

### 5.3 Hospital Database Test

The Hospital Database Test was done to ensure that the system was very efficient, user friendly and met all the set requirements needed for the project. It also ensures that the system meets the user expectations. As stated earlier, the users of the system are administrator, receptionist, nurses, doctors, lab technicians, and pharmacists. Hence, each user tested the system on their respective interfaces.

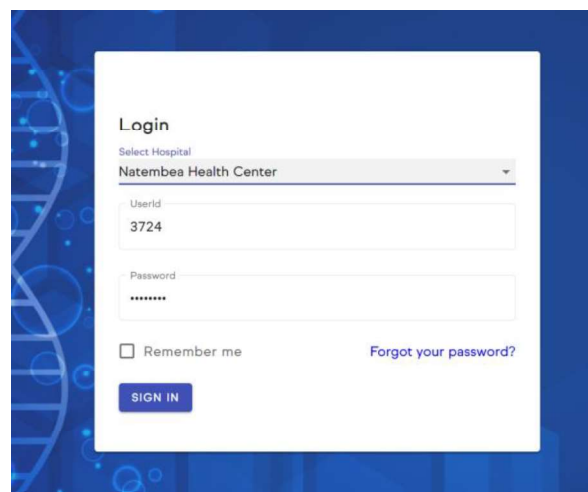


Figure 5.7: Login Interface

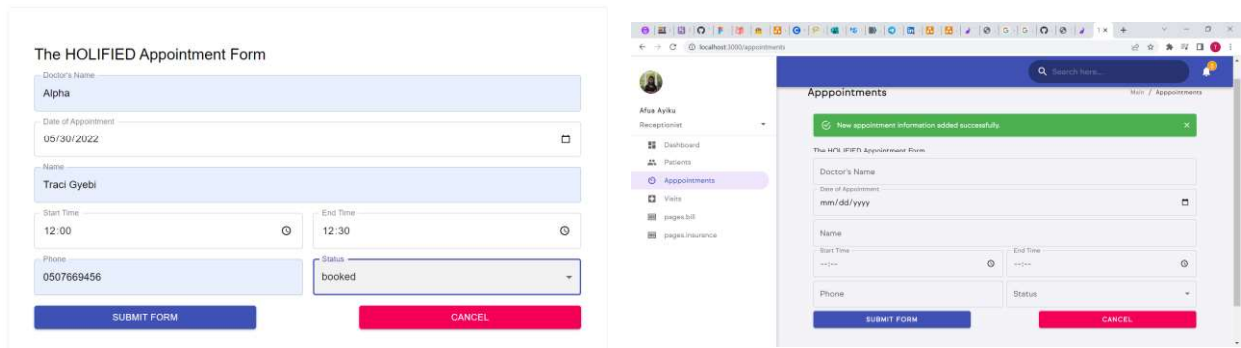


Figure 5.8: Appointment Form

### 5.3.1 Administration Test

The administrator was provided with the login details to the administration interface of the system. The administrator tested the key functionalities associated with the component including creating, reading, updating, and deleting users of the system, specialization, pharmacy, vaccine, insurance, viewing patient data requests from other hospitals, and assigning specializations to doctors.

The administrator's interface has a navigation bar which makes it easy for one to navigate through the various functionalities.

For generating the data for the total appointment per day, total patients per day, the average number of patients per day and the appointment status pie chart, the function aggregate was called in the appointment model.

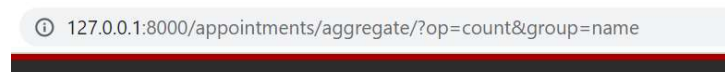


Figure 5.9: The URL for checking the aggregate of the operation and the field.

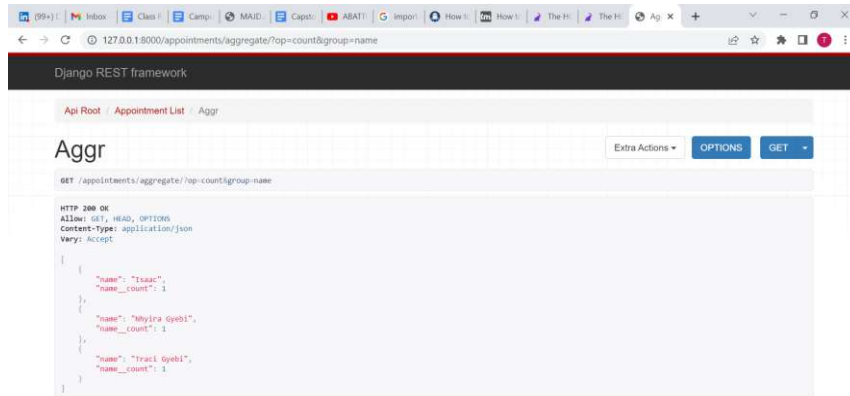


Figure 5.10: Successful aggregate function

The administrators used JSON format in registering the various users into the system. Below is a successful registration of a receptionist into the system. It follows the same process in assigning doctors to their specializations, and the other assignments.

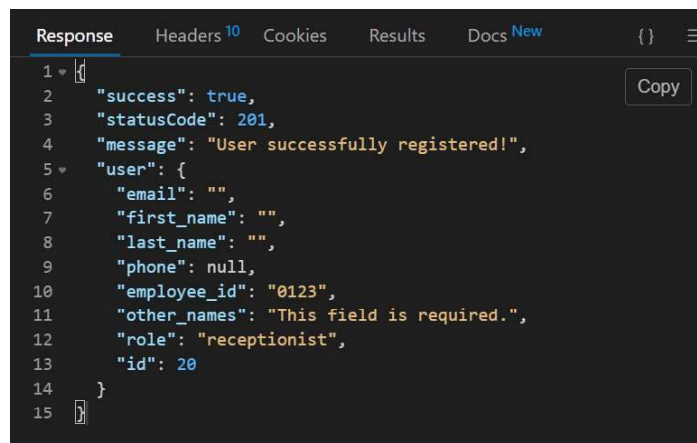


Figure 5.11: Successful registration

### 5.3.2 Receptionist Test

The receptionist was provided with the login details to the receptionist interface of the system. The receptionist tested the key functionalities associated with the component including creating, reading, updating, and deleting new patients, visits, bills, and insurance.

The receptionist interface has a navigation bar which makes it easy for one to navigate through the various functionalities.

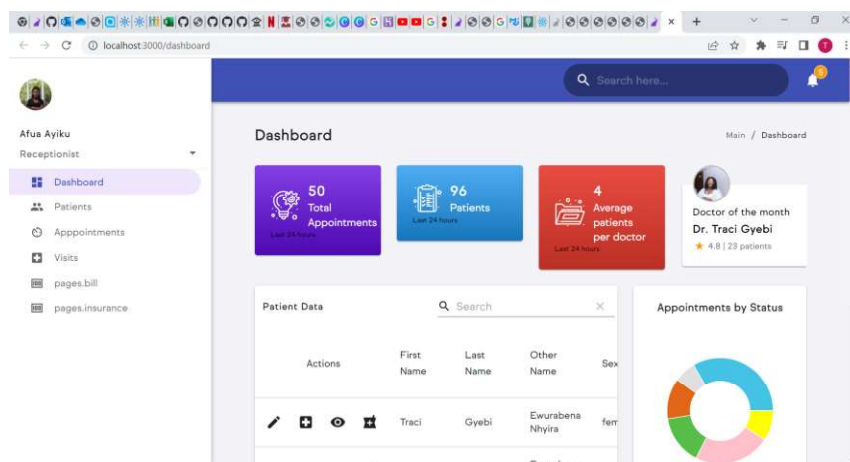


Figure 5.13: Receptionist Interface

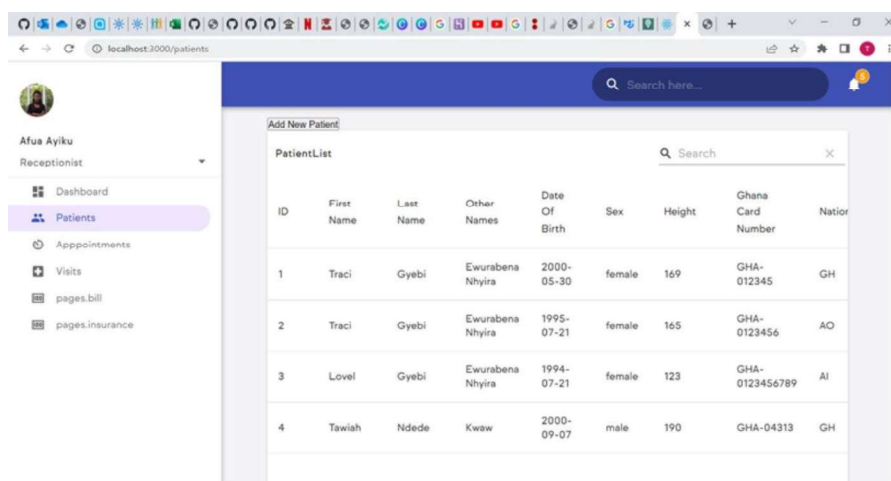


Figure 5.14: Patients form

The actions of each form can be found at the extreme end of the page, however, with regards to adding a new page, the user would have to click on “add new patient” at the top of the page.

### **5.3.3 Nurse Test**

The nurse was provided with the login details to the nurse interface of the system. The nurse tested the key functionalities associated with the component including creating, reading, updating, and deleting vitals, allergies, vaccines, assigning allergies and vaccines to patients, and adding patients through the Ghana Card medium.

The nurse interface has a navigation bar which makes it easy for one to navigate through the various functionalities. The nurse was able to record the oxygen level of the patient with ease with the help of the pulse oximeter. After which the data was transferred unto the database of the hospitals’ management system and popped up a message to advise the patient on his or respective habit.

The form with the patient’s vitals was designed to be simple to allow the nurses to input a range of vitals easily. Also, the table was designed in such a way that it was very easy to identify patients in the management system using the search feature.

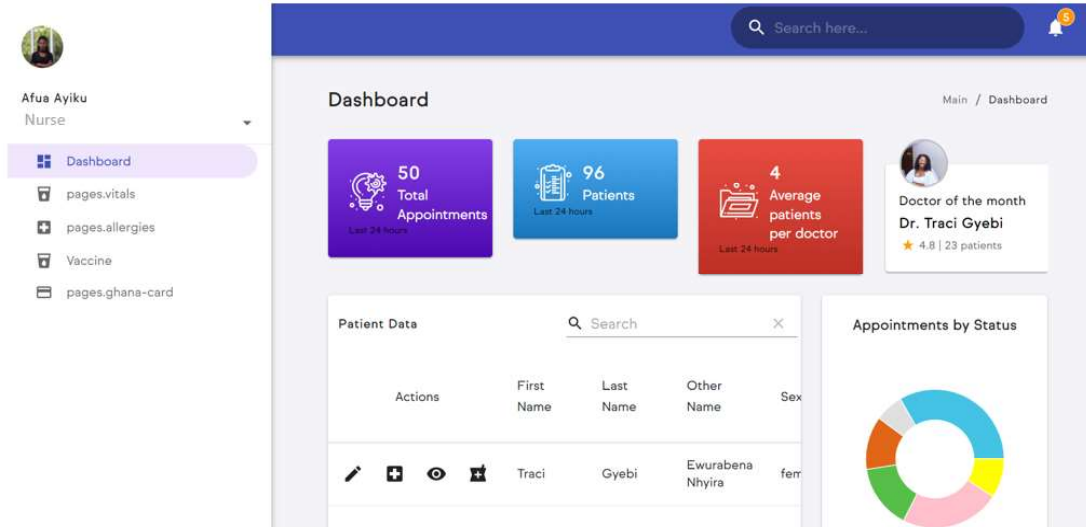


Figure 5.15: Nurse Interface

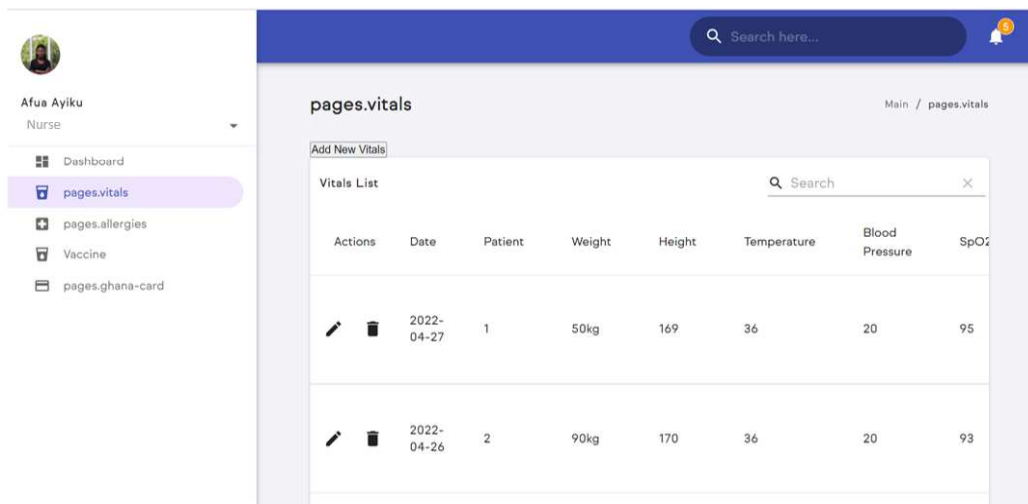


Figure 5.16: Vitals Form

### 5.3.4 Doctor Test

The doctor was provided with the login details to the doctor interface of the system. The doctor tested the key functionalities associated with the component including creating, reading, updating, and deleting records, prescriptions, schedules, assigning vaccines and allergies to patients and assigning drugs to their prescriptions.



The doctor's interface is very efficient and user friendly. It was modelled in such a way that doctors with little or no knowledge in computers will be able to navigate through the system since it was built for low resource hospitals.

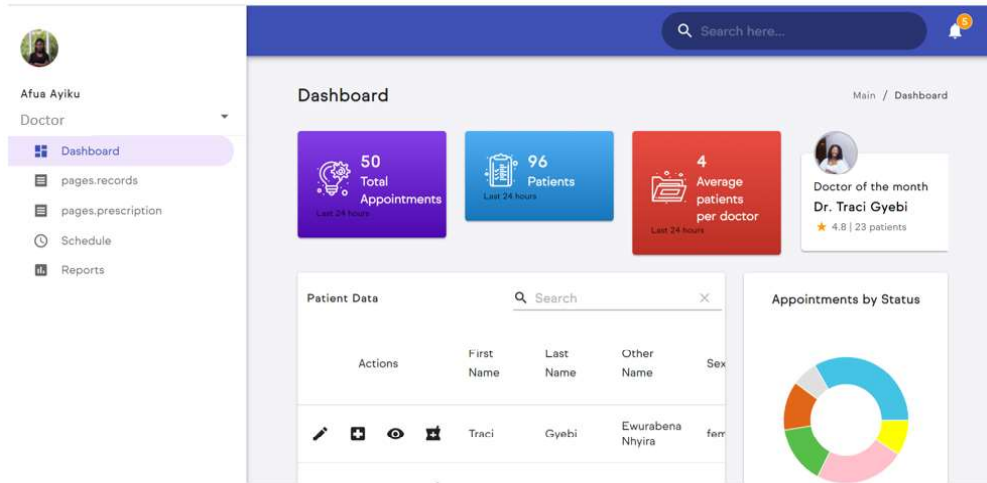


Figure 5.17: Doctor Interface

The image shows a form for scheduling appointments. It includes four input fields: 'Doctor' (with a dropdown menu showing '0001:'), 'Day' (with a dropdown menu showing 'monday'), 'Start time' (with a time picker), and 'End time' (with a time picker). A blue 'POST' button is located at the bottom right of the form.

Figure 5.18: Schedule Form (from the backend)

### 5.3.5 Lab Technician Test

The lab technician was provided with the login details to the lab technician interface of the system. The lab technician tested the key functionalities associated with the component including creating reading, updating, and deleting test types and lab test in relation to a particular visit.

The lab technician interface has a navigation bar which makes it easy for one to navigate through the various functionalities.

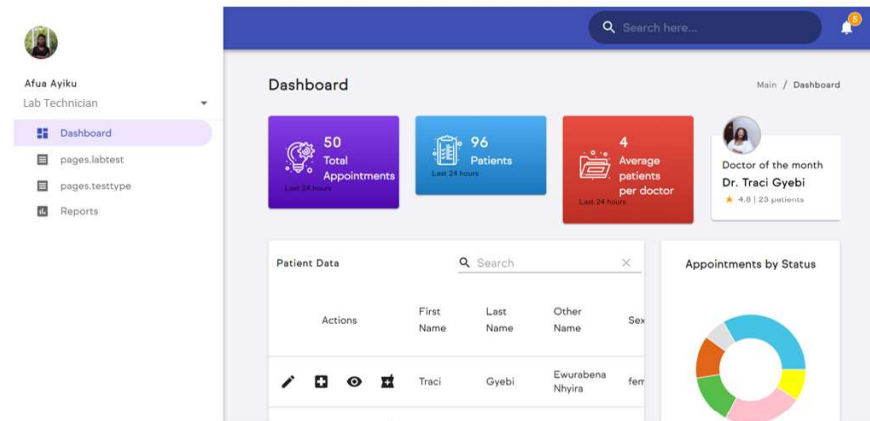


Figure 5.18: Lab Technician Interface

### 5.3.6 Pharmacist Test

The pharmacist was provided with the login details to the pharmacist interface of the system. The pharmacist tested the key functionalities associated with the component including viewing patient data, viewing medical prescription, and confirming the pickup of drugs.

The pharmacist interface has a navigation bar which makes it easy for one to navigate through the various functionalities.

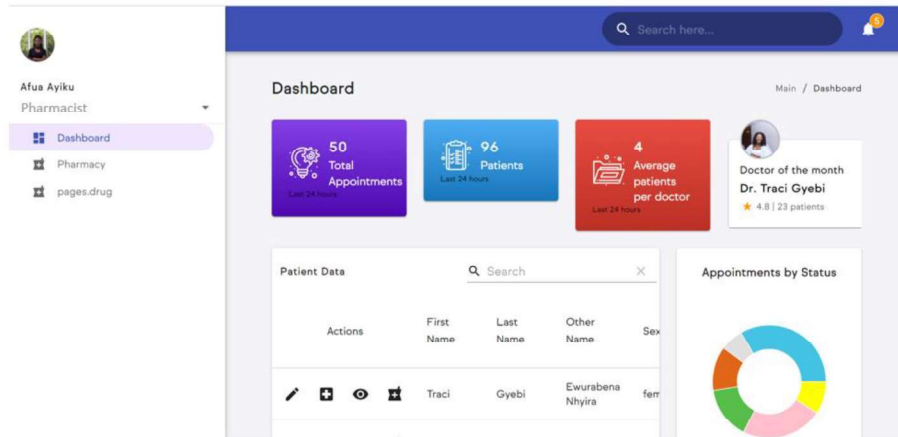


Figure 5.19: Pharmacist Interface

## 5.4 Pulse Oximeter Test

The pulse oximeter was compared to a real pulse oximeter that had already been developed about four times. This was done to ensure the pulse oximeter's effectiveness and accuracy.

When the two oximeter findings were compared, the newly constructed pulse oximeter was roughly 97% accurate.

Both pulse oximeters are depicted below:

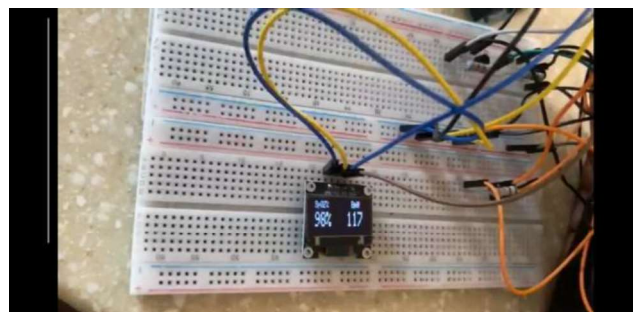


Figure 5.20: Built Pulse Oximeter



Figure 5.21: Original Pulse Oximeter

The data was successfully sent to the hospital management system's database. It appears on the front end of the website whenever new data is entered. A pop-up menu appears based on the range of the result.

A pop-up menu appears if the patient's oxygen saturation reading is between 95% and 100% — "Awesome! you are set to go!"

If the patient's oxygen saturation falls below 95%, a pop-up menu appears that says, "Hey there! Take care of yourself!"

```
SpO2: Yup.string().required('Please enter the accurate SpO2.').alertMessageType("Awesome, you are set to go!"),
```

Figure 5.22 The code for the pop up message that appears on full screen once the data from the pulse oximeter is entered into the system

---

**Awesome! You are set to go!**



Figure 5.23: Pop-up message

## **CHAPTER 6: Conclusion**

### **6.1 Overview**

This final chapter focuses on the challenges, future works, and conclusions of the project.

### **6.2 Challenges**

The major challenge encountered during the project was developing a system for a hospital's varied users, and given the project's tight deadline, it was critical to guarantee that the needs were not overburdened. Other difficulties encountered during the project were learning most of the languages required for the project within the project time frame. Finally, the initial microcontroller (Atmega8) intended for this project failed to work after several attempts due to the bootloader failing to install and the code failing to upload to the microcontroller, causing a delay in the pulse oximeter's construction.

### **6.3 Future Works**

The universal platform that will be used by low-resource hospitals currently cannot be accessed offline; therefore, if the project is to be continued, some changes should be made to accommodate an offline mode by using an offline database in conjunction with the hosted database, as well as other offline programming practices.

Due to time constraints, other vitals could not be recorded. As a result, if this project is to be completed, a full system capable of accurately detecting the weight and temperature of patients would be perfect.

Due to time constraints, scanning the Ghana card into the system to automatically fill the patients' form was also not possible. However, this could be improved in the future to help increase the system's efficiency and lessen the work burden on the receptionist, particularly if there is a long line.

Finally, the system has about six users: an administrator, a receptionist, a nurse, a doctor, a lab technician, and a pharmacist. If the project is continued, a patient-specific component should be considered, which would allow patients to log in and access details based on what the doctors allow under close monitoring, as the patient may currently only examine his or her records upon request.

#### **6.4 Conclusion**

The goal of this project is to focus on the internal operations of low-resource hospitals, to consolidate all institutions and allow them to share medical records, to improve patient communication, the speed with which medical records are transferred, and to improve patient satisfaction with the current system. The goal of the project is to also make it easier for Ghanaian healthcare providers to share patient data. By submitting a simple request through the web application, interested parties would have access to the client's information in the quickest time possible.

The project was divided into three main objectives. That is:

1. Hospital Database
2. Transfer of Patient Data
3. Pulse Oximeter

The goal of the Hospital Database is to make it easier to save essential information, not only about patients but also about the hospital, to improve its efficiency.

The goal of patient data transfer is to facilitate the rapid transfer of patient data from one hospital to another, particularly in an emergency. These hospitals, on the other hand, must use the same hospital management system.

Finally, in our current predicament, the pulse oximeter is an essential instrument. As a result, it is used to record patient oxygen levels as well as upload patient data to a database that can be examined by the nurse and will, in turn, display a pop-up message based on the patient's results.

By maintaining all information online and making incremental backups on all actions carried out, this system goes a long way toward eliminating some of the paper papers that would eventually pile up and prove ineffective.



## References

- [1] (2021). Retrieved 10 October 2021, from <https://www.carecloud.com/top-ehr-vendors/>
- [2] Advantages and Disadvantages of Electronic Health Records. (2021). Retrieved 10 October 2021, from <https://www.gallaghermalpractice.com/blog/post/advantages-and-disadvantages-of-electronic-health-records>
- [3] Electronic Medical Record Systems | AHRQ Digital Healthcare Research: Informing Improvement in Care Quality, Safety, and Efficiency. (2021). Retrieved 10 October 2021, from <https://digital.ahrq.gov/key-topics/electronic-medical-record-systems>
- [4] Pros and Cons of Paper Based Medical Records - TrueNorth ITG. (2021). Retrieved 10 October 2021, from <https://www.truenorthitg.com/pros-and-cons-paper-medical-records/>
- [5] What are the advantages of electronic health records? | HealthIT.gov. (2021). Retrieved 10 October 2021, from <https://www.healthit.gov/faq/what-are-advantages-electronic-health-records>
- [6] Institute of Medicine (US) Committee on Regional Health Data Networks, Donaldson, M. S., & Lohr, K. N. (1994). Confidentiality and privacy of personal data. National Academies Press.
- [7] (N.d.). Who.Int. Retrieved March 21, 2022, from [https://www.who.int/healthinfo/statistics/toolkit\\_hss/EN\\_PDF\\_Toolkit\\_HSS\\_InformationSystem\\_s.pdf](https://www.who.int/healthinfo/statistics/toolkit_hss/EN_PDF_Toolkit_HSS_InformationSystem_s.pdf)
- [8] Health information management history: Past, present & future. (n.d.). Rasmussen.Edu. Retrieved March 21, 2022, from <https://www.rasmussen.edu/degrees/health-sciences/blog/health-information-management-history/>
- [9] Net Health. (2021, September 16). The history of Electronic Health Records (EHRs). Net Health. <https://www.nethealth.com/the-history-of-electronic-health-records-ehrs/>
- [10] MacPherson, A. (1995). Nursing, health & the environment: Strengthening the relationship to improve the public's health. National Academies Press.
- [11] Careem, M., De Silva, C., De Silva, R., Raschid, L., & Weerawarana, S. (2006). Sahana: Overview of a disaster management system. 2006 International Conference on Information and Automation.
- [12] Institute Of Committee on Engineering and the Health Care System. (2005). Building a better delivery system: A new engineering/health care partnership (P. P. Reid, W. D. Compton, J. H. Grossman, & G. Fanjiang, Eds.). National Academies Press.
- [13] Health resources and services availability monitoring system (HeRAMS). (n.d.). Who.Int. Retrieved March 21, 2022, from <https://www.who.int/initiatives/herams>
- [14] Matengo, W., Otsieno, E., & Wanjiru, K. (2020). Big Data Analytics in Healthcare. In Digital Health in Focus of Predictive, Preventive and Personalised Medicine (pp. 129–138). Springer International Publishing.

[15] Darko, K. (2021, November 3). Health records at TTH can be monitored at Korle Bu, no need for physical folders - Bawumia on digitisation. MyJoyOnline.Com. <https://www.myjoyonline.com/health-records-at-tth-can-be-monitored-at-korle-bu-no-need-for-physical-folders-bawumia-on-digitisation/>

[16] Management of patient information: Trends and challenges in Member States. (n.d.). WHO |Regional Office for Africa. Retrieved April 25, 2022, from <https://www.afro.who.int/publications/management-patient-information-trends-and-challenges-member-states>

[17] Who Regional Office for the Western Pacific. (2006). Electronic health records: A manual for developing countries. World Health Organization.

[18] Abdekhoda, M., Dehnad, A., & Zarei, J. (2019). Determinant factors in applying electronic medical records in healthcare. *La Revue de Sante de La Mediterranee Orientale [Eastern Mediterranean Health Journal]*, 25(1), 24–33. <https://doi.org/10.26719/emhj.18.007>

[19] L. Poissant, J. Pereira, R. Tamblyn, and Y. Kawasumi, “The impact of electronic health records on time efficiency of physicians and nurses: a systematic review,” *Journal of the American Medical Informatics Association: JAMIA*, 2005. [Online]. Available: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1205599/>. [Accessed: 08-Nov-2020].

[20] Getting Started Guide. (n.d.). Mailtrap.Io. Retrieved April 27, 2022, from <https://help.mailtrap.io/article/12-getting-started-guide>